



**EnCompass Knowledge Systems, Inc.**

*Visualizing the Present, Designing the Future,  
Managing the Transition<sup>sm</sup>*

# **The EnCompass<sup>®</sup> Emergent Structure Analysis**

29 August 2002

**EXHIBIT A**

Title: Business Analysis & Management Systems  
Utilizing Emergent Structures

Inventors: Michael M. Mann & Arne Haugland

Attys.: Fulwider Patton et al. Dkt. # 65567/ENCMP

# TABLE OF CONTENTS

<b>1.0 INTRODUCTION .....</b>	<b>1</b>
<b>1.1 ENCOMPASS PROCESS OVERVIEW .....</b>	<b>1</b>
<b>1.2 THE ORGANIZATION VIEW .....</b>	<b>4</b>
<b>1.3 EMERGENT STRUCTURE: THE ISSUE VIEW.....</b>	<b>5</b>
<b>1.4 OUTPUT OPTIONS .....</b>	<b>6</b>
1.4.1 Combining emergent structures: meta-issues .....	7
1.4.2 Isolating duplicate nodes .....	8
1.4.3 Showing paths between nodes .....	9
<b>2.0 THE EMERGENT STRUCTURE ANALYSIS .....</b>	<b>11</b>
<b>2.1 KEY CONCEPTS AND DEFINITIONS .....</b>	<b>11</b>
<b>2.2 DATA OBJECT DEFINITIONS .....</b>	<b>14</b>
2.2.1 <i>ANR</i> : Analysis definition object .....	14
2.2.2 <i>DIE</i> : Data collection record object .....	16
2.2.3 <i>DIG</i> : Data collection link object.....	16
2.2.4 <i>ISR</i> : Issue record object .....	17
2.2.5 <i>ENE</i> : Emergent nomination element.....	17
2.2.6 <i>ESE</i> : Emergent structure element .....	17
2.2.7 <i>CNC</i> : Connection object .....	17
<b>2.3 TOP-LEVEL PROGRAM STRUCTURE .....</b>	<b>18</b>
2.3.1 Program flow .....	18
2.3.2 Diagram 1: Overall program structure.....	20

---

<b>2.4 PROGRAM FLOW DIAGRAMS .....</b>	<b>22</b>
<b>2.5 DETAILED DISCUSSION .....</b>	<b>37</b>
2.5.1 Introduction .....	37
2.5.2 Diagram 2: Retrieving data collection records .....	37
2.5.3 Diagram 3: Creating data collection links .....	39
2.5.4 Diagram 4: Merging data collection models.....	41
2.5.5 Diagram 5: Listing master issues .....	45
2.5.6 Diagram 6: Creating emergent structure elements.....	46
2.5.7 Diagram 7: Nomination.....	47
2.5.8 Diagram 8: Inserting slave issues.....	48
2.5.9 Diagram 9: Creating connection records .....	50
2.5.10 Diagram 10: Deleting existing connections .....	51
2.5.11 Diagram 11: Inserting new connections.....	51
<b>3.0 DISPLAY OPTIONS .....</b>	<b>53</b>
<b>3.1 REPORT GENERATORS .....</b>	<b>53</b>
<b>3.2 CUSTOM DISPLAY APPLICATIONS .....</b>	<b>53</b>
3.2.1 Hierarchical data display libraries .....	54
3.2.2 3D display engines.....	54
3.2.3 The Parasol application and display engine .....	55
<b>4.0 EMERGENT VIEW EXAMPLE .....</b>	<b>57</b>

## LIST OF FIGURES

Figure 1: Project issues.....	2
Figure 2: EnCompass data collection instrument .....	3
Figure 3: Data collection record—General data.....	3
Figure 4: Data collection record—Issue-specific data .....	3
Figure 5: Data collection links .....	4
Figure 6: Data collection links with organization structure .....	5
Figure 7: Emergent structure.....	6
Figure 8: Meta-issue .....	7
Figure 9: Duplicate nodes .....	8
Figure 10: Closeness .....	9
Figure 11: Closeness with shortest path isolated .....	10
Figure 12: Emergent structure objects .....	18
Figure 13: Model 1 query selection .....	20
Figure 14: <i>Show Results</i> panel .....	41
Figure 15: Master and slave issue tree.....	49
Figure 16: Replace existing connections .....	51
Figure 17: Parasol application structure .....	56
Figure 18: Emergent structure display, 3D view .....	58
Figure 19: Emergent structure display, outline view .....	59

---

## PROGRAM FLOW DIAGRAMS

Diagram 1: Emergent Structure Analysis.....	19
Diagram 2: Retrieving data collection records .....	23
Diagram 3: Creating data collection links .....	25
Diagram 4: Merging data collection models.....	26
Diagram 5: Listing master issues .....	30
Diagram 6: Creating emergent structure elements .....	31
Diagram 7: Nomination .....	32
Diagram 8: Inserting slave issues.....	33
Diagram 9: Creating CNC objects .....	34
Diagram 10: Deleting existing connections.....	35
Diagram 11: Inserting new connections.....	36

# The EnCompass Emergent Structure Analysis

## 1.0 INTRODUCTION

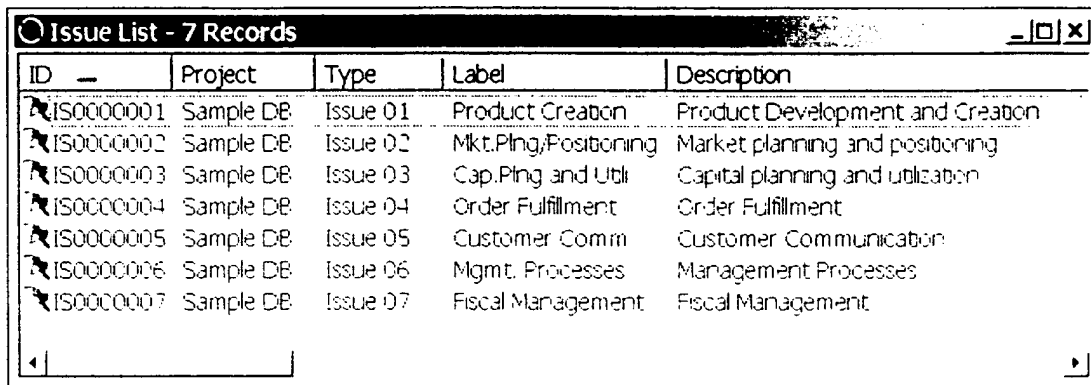
The emergent structure analysis forms part of the EnCompass system, a computer-supported tool for enterprise process analysis and change management. The system does the following:

- collects data on the patterns of interaction among people in an organization
- analyzes these relationships with respect to specific organizational processes and issues
- presents the results in a graphical format

## 1.1 ENCOMPASS PROCESS OVERVIEW

The following broad outline characterizes the main steps in the EnCompass analysis process:

**1. Definition of study issues and participants**—EnCompass consultants meet with organization team members to determine the problems to be addressed and the specific *issues* that are likely to be relevant to the study (fig. 1).



The screenshot shows a window titled "Issue List - 7 Records" with a table containing 7 rows of project issues. Each row has a tree icon in the first column. The table columns are ID, Project, Type, Label, and Description.

ID	Project	Type	Label	Description
IS0000001	Sample DB	Issue 01	Product Creation	Product Development and Creation
IS0000002	Sample DB	Issue 02	Mkt.Plng/Positoning	Market planning and positioning
IS0000003	Sample DB	Issue 03	Cap.Plng and Utli	Capital planning and utilization
IS0000004	Sample DB	Issue 04	Order Fulfillment	Order Fulfillment
IS0000005	Sample DB	Issue 05	Customer Comm	Customer Communication
IS0000006	Sample DB	Issue 06	Mgmt. Processes	Management Processes
IS0000007	Sample DB	Issue 07	Fiscal Management	Fiscal Management

Figure 1: Project issues

**2. Data collection**—EnCompass creates a *data collection instrument* (fig. 2) to gather information on organization members' interactions with each other. Study participants indicate the specific individuals they interact with on a regular basis, and rate the overall frequency and importance of those interactions and their impact on each of the specific issues that were identified in step 1, on a scale of 0 to 5. The exact meanings of these values may vary from study to study; typical ranges are from "less than once a month" (1) to "several times a day" (5) for frequency, "not important" (1) to "critically important" (5) for importance, and "seldom" (1) to "always" for impact. (Zero represents no interaction.) In addition to the frequency, importance, and impact values, survey respondents may also be asked to estimate the number of hours per week spent per week interacting with each person on the list, and report the primary *modus* of these interactions: whether they are usually *concurrent* (face-to-face meetings, telephone, etc.) or *nonconcurrent* (email, memos, etc.).

Name John Frees ID# 103

With whom do you interact at least monthly in ways that are generally important in order to get our tasks done, or to help the other person get his/her tasks done?

Frequency Range: 1 = Monthly, 2 = Bi-monthly, 3 = Quarterly, 4 = Semi-annually, 5 = Annually  
Importance Range: 1 = Not at all, 2 = A little, 3 = Fairly, 4 = Quite a bit, 5 = Very much

Impact Range: 1 = Not at all, 2 = A little, 3 = Fairly, 4 = Quite a bit, 5 = Very much

Issue	Frequency	Importance	Impact
D. Sales	4	4	4
D. Budget	3	3	3
P. Policy	3	3	3
P. Data	2	2	2
N. Data	1	1	1
J. Scheduling	2	2	2
P. Product	3	3	3
C. Data	3	3	3

Figure 2: EnCompass data collection instrument

Each line in the completed data collection instrument represents a regular interaction between the survey respondent and one other specific organization member. For each regular interaction, the respondent has estimated its overall frequency and importance and its impact on each of the identified study issues. When the survey information is entered into the EnCompass database, each line becomes a single *data collection record* (figs. 3, 4).

General | Issues 1-10 | Issues 11-20 |

Id: DI0000128

Project: Sample DB

Type: As Is

Subtype:

Label:

Survey Date: 10/22/2001

Person Interaction

Frequency: 4 Importance: 4

Interaction

From Person: John Frees

To Person: Dick Scales

Figure 3: Data collection record—General data

General | Issues 1-10 | Issues 11-20 |

From: John Frees To: Dick Scales

Product Creation

Impact: 4 Frequency: 0 Hours per week: 0.00

Mkt.Ping/Positioning

Impact: 5 Frequency: 0 Hours per week: 0.00

Cap Ping and Util.

Impact: 0 Frequency: 0 Hours per week: 0.00

Order Fulfillment

Impact: 3 Frequency: 0 Hours per week: 0.00

Customer Comm.

Impact: 5 Frequency: 0 Hours per week: 0.00

Mgmt. Processes

Impact: 4 Frequency: 0 Hours per week: 0.00

Fiscal Management

Impact: 4 Frequency: 0 Hours per week: 0.00

Figure 4: Data collection record—Issue-specific data



**3. Data analysis**—Using the EnCompass software, the consultant creates *analyses*, examining the interaction patterns of selected combinations of people and issues. For each analysis, the software converts the data to a three-dimensional graphical display (fig. 5). By interpreting the display and creating additional analyses to examine specific types of interaction, the analyst locates areas where decision-makers are not well integrated into the organizational structure, or where mismatched perceptions and inefficient communications channels may be interfering with the accomplishment of the organization's process objectives.

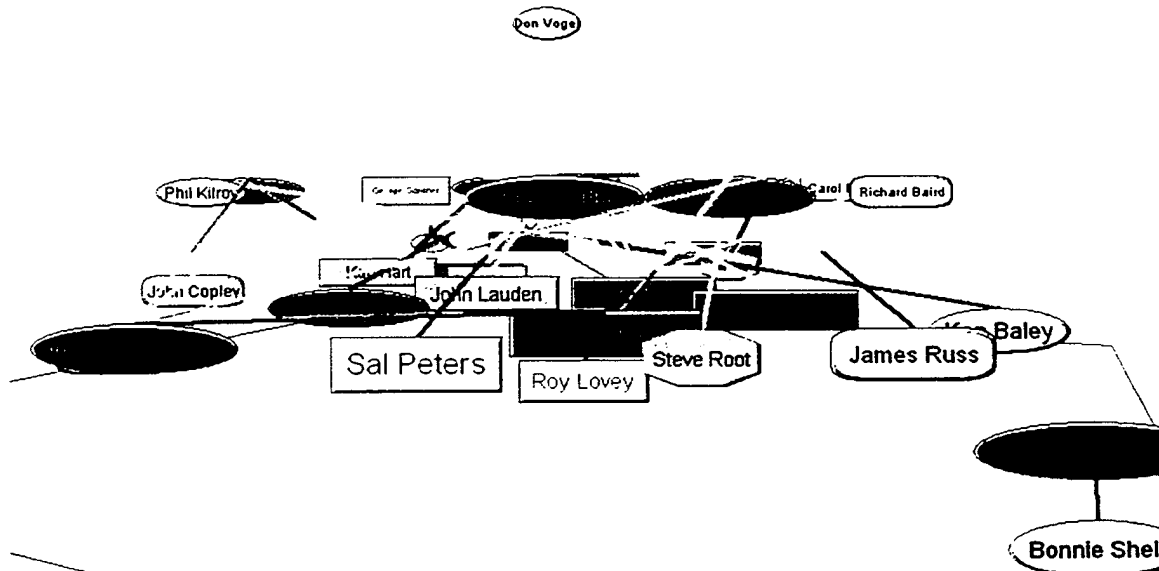


Figure 5: Data collection links

**4. Report and recommendations**—EnCompass consultants meet again with the organization team to review their findings, discuss opportunities for improvements, and make final recommendations.

## 1.2 THE ORGANIZATION VIEW

Figure 5 above illustrates the display of interaction links between members of an organization. What is missing from the picture is the structural context which gives meaning to these patterns of interaction and communication. This view of the data is an *organization view*, in which the context against which the links are displayed is the

traditional hierarchy of job titles and formal organizational reporting structure. When made visible, this implicit context appears as shown in figure 6. Here, an individual's vertical position depends on his or her standing in the organization's formal structure.

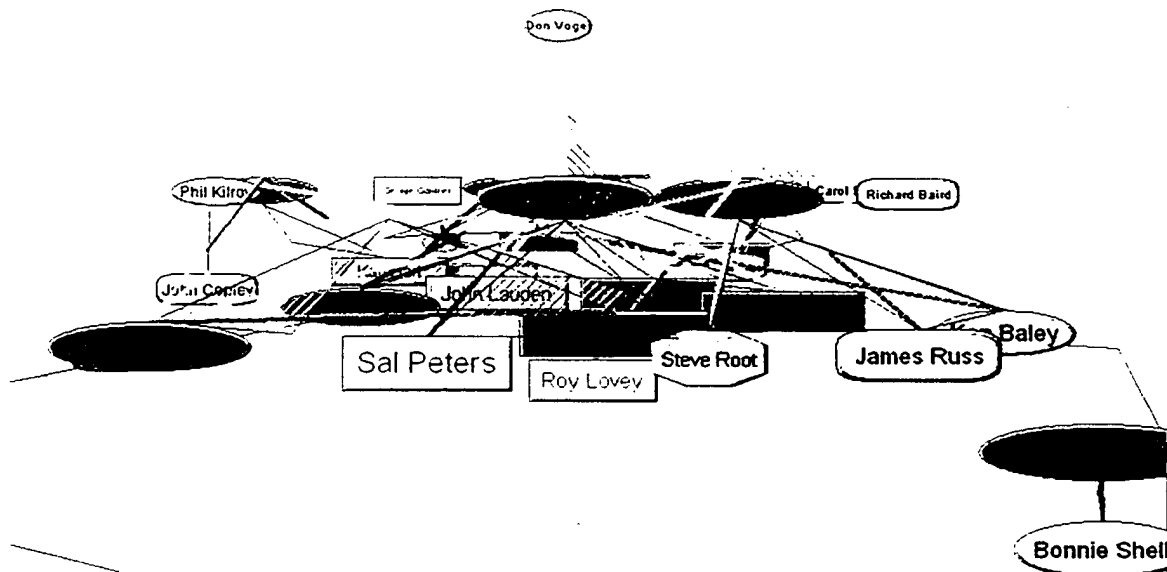


Figure 6: Data collection links with organization structure

### 1.3 EMERGENT STRUCTURE: THE ISSUE VIEW

In addition to the organization view described in section 1.2, EnCompass provides a second method of visualizing interaction patterns, called the *issue view*. Rather than a fixed, formal hierarchy, this view displays interactions between members of an organization against the background of an *emergent* structure: that is, one which *arises from the significance that individual members assign to one another*—specifically, their perceived *impact* on a given issue. In this view, a person's status may be quite different from his position in the organization's formal hierarchy (fig. 7, page 6). This view provides valuable insights into the real communication and decision-making patterns within an organization.

The **emergent structure analysis** is the set of procedures that creates these structures. (The term "create" is used here to mean the process of deriving these emergent structures and their associated issue views from the tabular data in the database.)

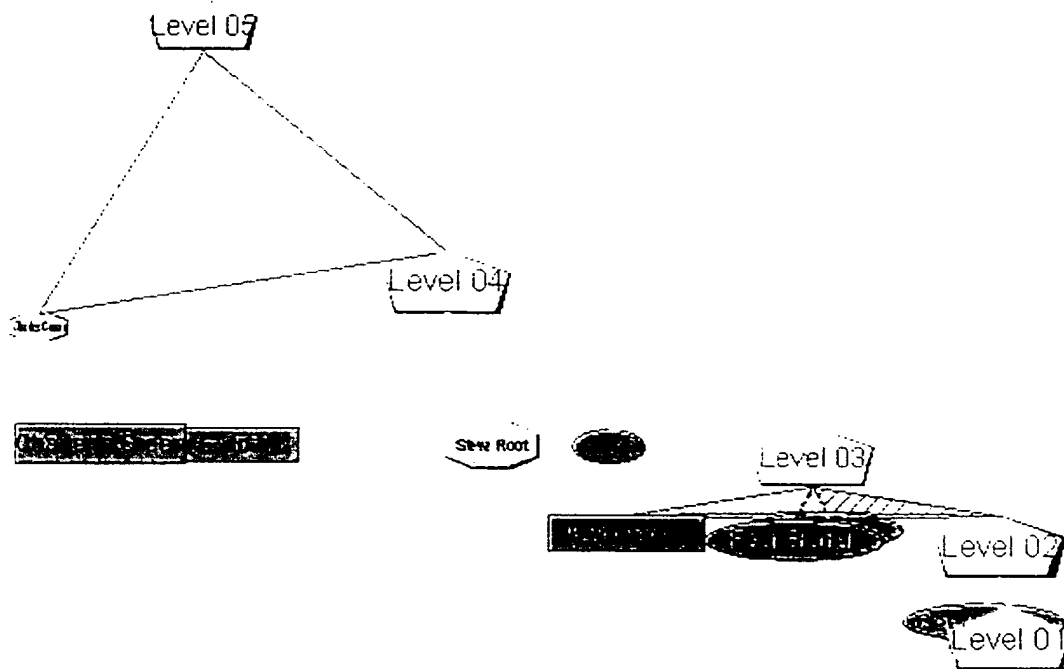


Figure 7: Emergent structure

Section 2 of this document (page 11) describes the emergent structure analysis process.

## 1.4 OUTPUT OPTIONS

Once the program has created an emergent structure and displayed it in a tree view, other existing components of the EnCompass software enable the analyst to perform additional manipulations of the displayed data.

- Combine emergent structures under meta-issues (section 1.4.1, page 7)
- Isolate duplicate nodes (section 1.4.2, page 8)
- Show the paths between nodes (section 1.4.3, page 9)

### 1.4.1 Combining emergent structures: meta-issues

The standard issue view presents a "virtual hierarchy" which emerges around a single issue: people in the organization appear at different levels depending on their perceived

impact on that issue. It is often useful, however, to view and compare the emergent structures of more than one issue at the same time. For example, the analyst might want to see who are a company's important decision-making figures in the arenas of both sales and marketing, and investigate the communications links between those groups. To include multiple issues in the same view, EnCompass provides the ability to create **meta-issues**.

When the analyst wishes to study multiple issues, he or she creates a meta-issue entry in the database and manually connects the related *master issues* to it. The resulting display is illustrated in figure 8.

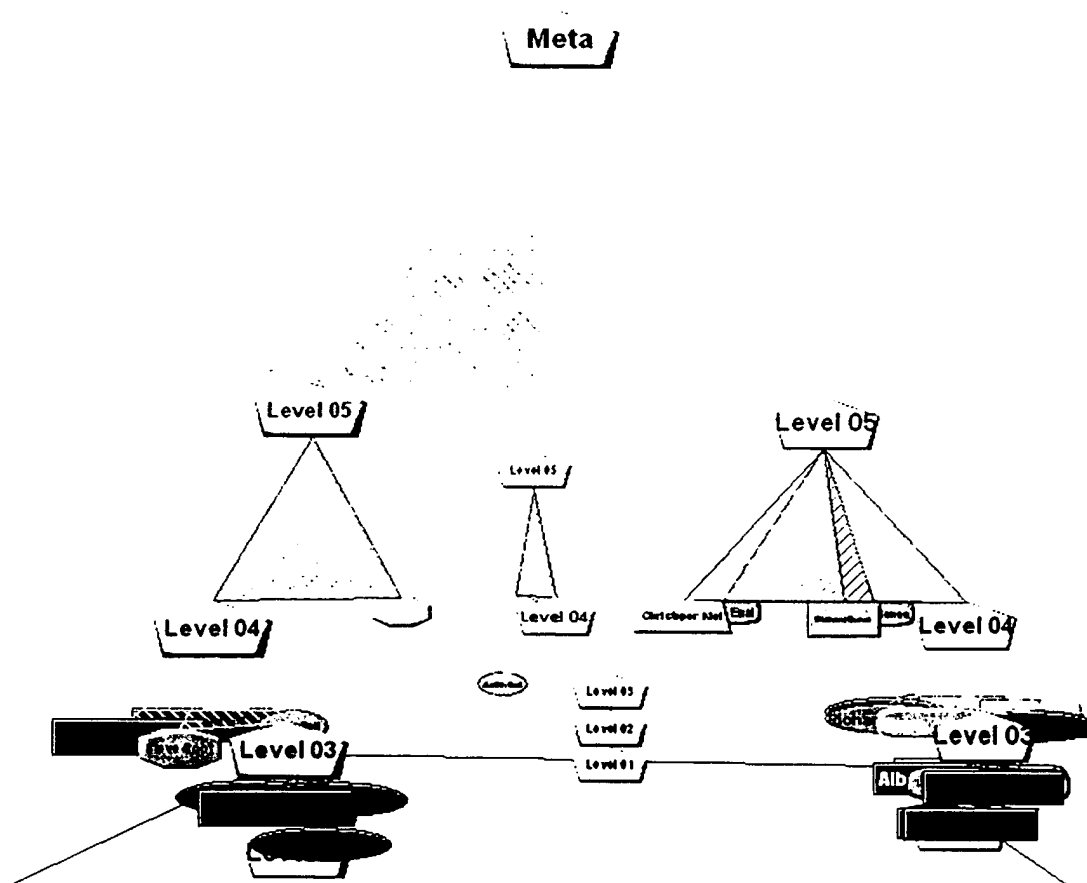


Figure 8: Meta-issue

### 1.4.2 Isolating duplicate nodes

The *Show Duplicate Nodes* function isolates any objects that appear more than once in the issue view display. This feature is useful when examining multiple-issue analyses involving meta-issues, where it is often important to verify that decision-making processes are well integrated across related issues. For example, the same decision-makers may need to be involved in both product design and cost planning. Conversely, in other cases it may be advisable to ensure that the same individuals do not play a role in areas that should remain separate, such as corporate auditing and investment management.

Figure 9 shows the result of the *Show Duplicate Nodes* function when applied to the meta-issue display in figure 8.

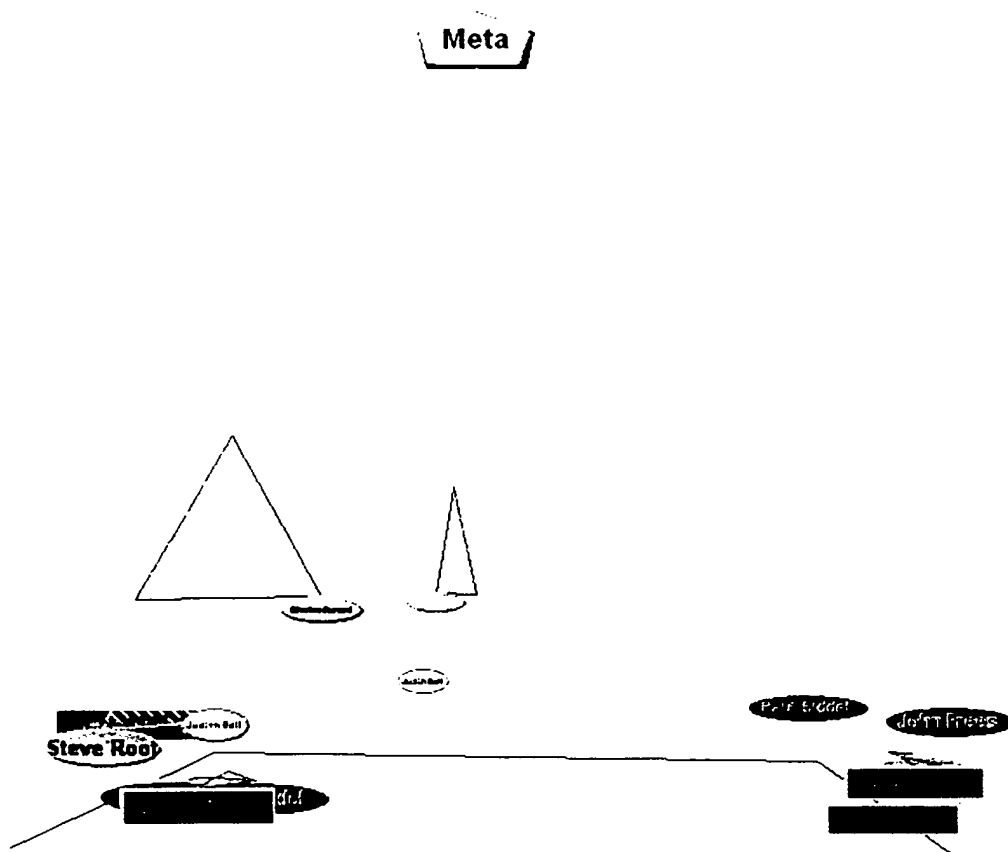


Figure 9: Duplicate nodes

### 1.4.3 Showing paths between nodes

When studying communication patterns among a group of people, an important concept is that of *closeness*: the distance between any two individuals, as expressed by the number of nodes between them. For the analyst, this measure reveals how closely two specific members of an organization are linked on a given issue—whether they communicate with each other directly, indirectly through intermediaries, or not at all. Unnecessary steps in the process can represent opportunities for information to be delayed or distorted, whereas direct communication that inappropriately bypasses the chain of command and leaves others "out of the loop" can subvert the aims of the organizational structure.

In many cases, a complex or crowded display can obscure the links between specific people. Furthermore, they may be connected by multiple routes, some more direct than others, so that it becomes difficult to locate the shortest path between them. To find this information, EnCompass provides the *Closeness* function, which automatically calculates and displays the shortest path between any two nodes in the tree.

Figure 10 illustrates the display with the closeness path between two nodes.

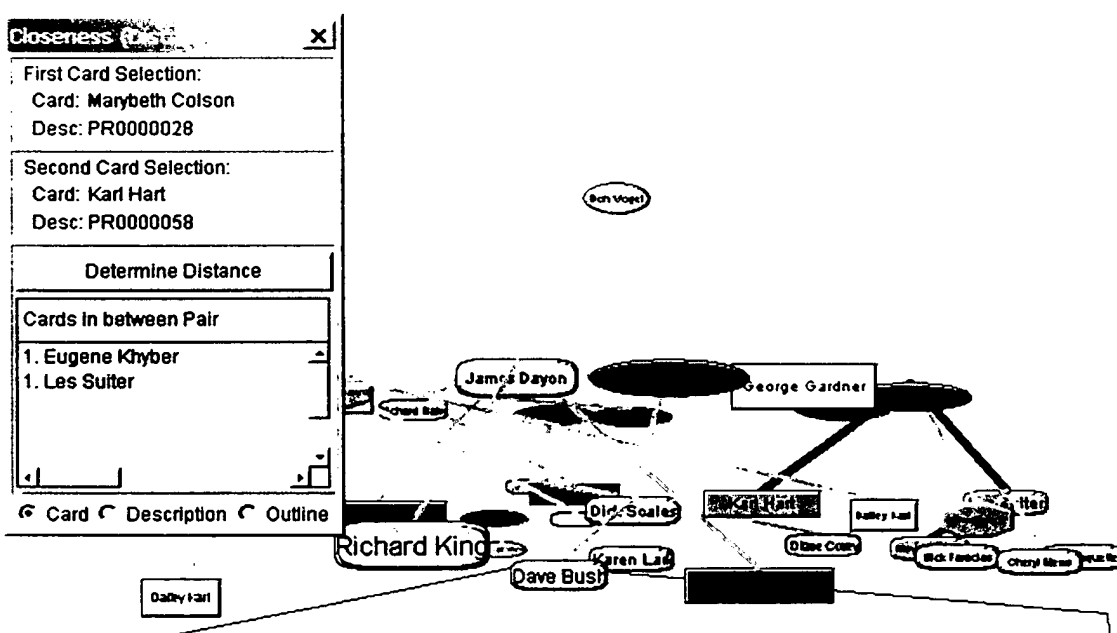


Figure 10: Closeness

Once the program has determined the shortest path between two nodes, the analyst can elect to subtract all other links from the display, isolating the desired path (fig. 11).

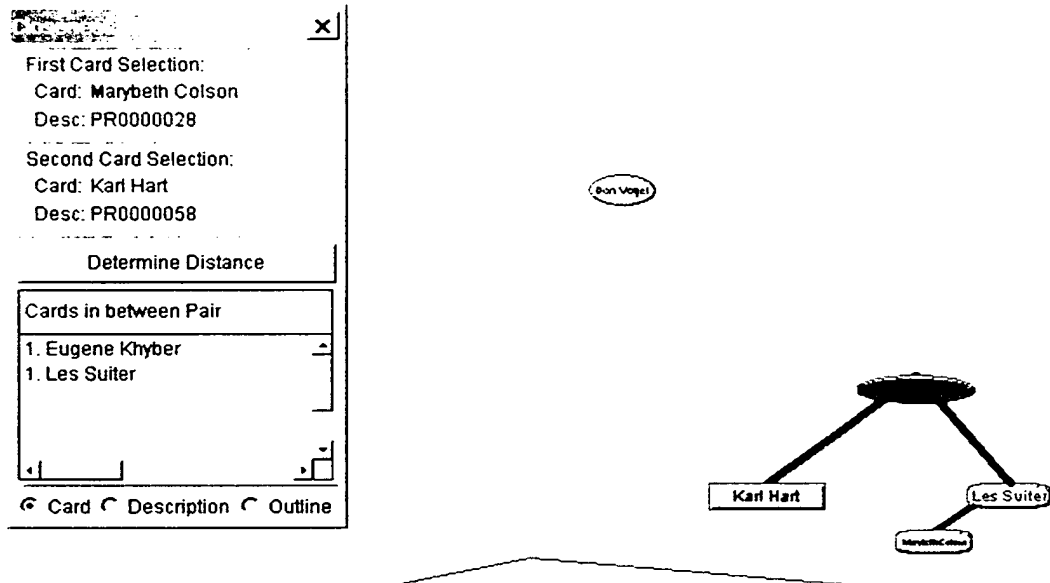


Figure 11: Closeness with shortest path isolated

## 2.0 THE EMERGENT STRUCTURE ANALYSIS

The emergent structure analysis is a selection and transformation process. Its input data are the issues that have been defined as important for an organizational study and the organization members' reported perceptions of their regular interactions with others about those issues, as gathered by the data collection instrument (page 3). According to criteria defined by the analyst (using the EnCompass software), the process selects a subset of this raw data, creates a suitable structure, and connects each person record to its appropriate level in the structure.

### 2.1 KEY CONCEPTS AND DEFINITIONS

Our description of the emergent structure analysis employs the following concepts:

The **project** (or "study") is the large-scale unit of organization for work in EnCompass, typically involving the analysis of interactions among members of a single company or other organization. For each project, the project designer identifies the relevant *issues* to be examined, and creates a data collection instrument to measure interactions based on those quantities.

**Issues** are the concerns that drive decisions and activities within an organization, typically involving business processes such as "market planning" or "customer communication." For each study, the EnCompass consulting team works with representatives from the organization to identify the specific issues that need to be examined. The EnCompass *data collection instrument* measures the impact of interactions between individuals with respect to every one of the identified issues; these values are stored in individual *data collection records*.

- **Master issues** are basic records that exist for every issue in every study, as illustrated in figure 1 (page 2). There is one master issue for each issue defined in a study.
- **Slave issues** are created when needed for an emergent analysis. To each master issue that is examined in an emergent analysis, the system attaches six identically named slave issues, labeled respectively "Level 5" through "Level 0."



**Data collection records** contain information on the regular interactions between members of an organization in an EnCompass study, as gathered by the study's survey questionnaire. Each data collection record contains the rankings that a specific person (the "From" person) gave to his or her interaction with one other person (the "To" person), in terms of its overall frequency and importance, and its impact on each of the *issues* that the project designer has defined as significant (see figures 3 and 4, page 3). Valid values for frequency, importance, and impact are integers from 0 to 5.

A **data collection link** is a connection between a survey respondent (the "From" person) and another member of the organization (the "To" person), representing a regular interaction which the respondent judges to have some importance to his ability to accomplish his work-related tasks.

- Links may be either **confirmed** or **unconfirmed**. A *confirmed* link is one that goes in both directions; that is, the interaction is reported by both parties. Every confirmed link, therefore, consists of two separate data collection records: one reflecting the interaction  $A \rightarrow B$ , the other  $B \rightarrow A$ . (Person A and person B may, however, make very different estimates of the interaction's frequency, importance, and impact.) An *unconfirmed* link is one that is reported by only one of the parties:  $A \rightarrow B$  but not  $B \rightarrow A$ . A high percentage of unconfirmed links within an organization indicates disagreement or lack of communication about its processes.
- The EnCompass display engine converts these links to a graphical form for display and interpretation (see figure 5, page 4). Each link appears on a three-dimensional tree diagram as a line between two nodes, representing the two parties to the interaction. The width and color of the lines indicate frequency, importance, and issue-impact values. The analyst chooses which links to display by selecting one of the user-defined analysis definitions.

The **analysis definition** contains detailed information on the parameters that determine which *data collection links* are included in a given analysis. Each analysis selects *data collection records* from the database according to a number of *SQL queries*, constructed by the analyst using the EnCompass software interface.

**SQL queries** are statements that request specific information from a database, using the standard *Structured Query Language*. For example, the following SQL query:

```
SELECT last_name, first_name FROM employees WHERE age < 50 ORDER
BY last_name
```

would retrieve the last name and first name of all employees whose age is less than 50, sorting the output by last name.

Each EnCompass analysis selects *data collection records* for display based on the following types of user-defined queries, distinguished by the database location of the information they are searching for:

- **Data collection** queries look for information contained in the data collection records themselves, such as interaction frequency, importance, and impact. At least one data collection query is required for every analysis.
- **Person** queries select records based on information in the individual person records, such as corporate title, function, salary, or tenure. Person queries are optional when setting up an analysis.
- **Organization** queries search for records based on information about the organization, such as address or other defined attributes. Organization queries are optional.
- **Criteria** are ranges of values within which the two parties to an interaction must agree or disagree in order for the data collection record to be included in the current analysis. Criteria queries typically contain complex combinations of parameters. For example, an analyst might wish to view only those interactions where the impact values agree within a range of  $\pm 2$  on the issue being examined, and within a range of  $\pm 5$  on all other issues.

**Model 1 and model 2:** An analysis definition may contain a second complete set of search parameters, representing a second set of *SQL queries*. The database search engine retrieves the data collection records matching both of these query sets, and the analyst can elect to add the results together, subtract one from the other, or merge the results according to specific conditions.

NOTE: In this discussion, the number of alternative analysis models is limited to two, because the existing EnCompass software only makes use of this number. The emergent structure process itself, however, can incorporate additional models, if a suitable user interface were constructed to accommodate them.

**Nomination** refers to the number of survey respondents who must assign a given impact value to their interactions with a person for that person to rise to a given level in the emergent structure. For example, if the nomination level in an analysis is set to 10, person B will be promoted to level 5 if and only if at least 10 other people assign an impact value of 5 to their interactions with person B. People who fail to receive the specified number of nominations at any level appear at level 0.

## 2.2 DATA OBJECT DEFINITIONS

In order to transform raw data collection records into connections between persons and numbered slave issues, the emergent structure analysis creates and manipulates a number of data objects (classes), consisting of pointers to columns in the database. Some of these objects correspond closely to the structure of database records. Through a series of comparisons, these are successively merged and transformed into new objects containing the desired data.

### 2.2.1 ANR: Analysis definition object

<i>ID</i>	<i>Project</i>
<i>Type</i>	<i>Status</i>
<i>Analysis Name</i>	<i>Description</i>
<i>Model 1 Criteria</i>	<i>Model 2 Criteria</i>
<i>Model 1 Criteria ID</i>	<i>Model 2 Criteria Id</i>
<i>Model 1 Dir1 Name</i>	<i>Model 2 Dir1 Name</i>
<i>Model 1 Dir1 Type</i>	<i>Model 2 Dir1 Type</i>
<i>Model 1 Dir1 Person ID</i>	<i>Model 2 Dir1 Person ID</i>
<i>Model 1 Dir2 Name</i>	<i>Model 2 Dir2 Name</i>

<i>Model 1 Dir2 Type</i>	<i>Model 2 Dir2 Type</i>
<i>Model 1 Dir2 Person ID</i>	<i>Model 2 Dir2 Person ID</i>
<i>Model 1 From Person Name</i>	<i>Model 2 From Person Name</i>
<i>Model 1 From Person Type</i>	<i>Model 2 From Person Type</i>
<i>Model 1 From Person ID</i>	<i>Model 2 From Person ID</i>
<i>Model 1 From Organization Name</i>	<i>Model 2 From Organization Name</i>
<i>Model 1 From Organization Type</i>	<i>Model 2 From Organization Type</i>
<i>Model 1 From Organization Person ID</i>	<i>Model 2 From Organization Person ID</i>
<i>Model 1 To Person Name</i>	<i>Model 2 To Person Name</i>
<i>Model 1 To Person Type</i>	<i>Model 2 To Person Type</i>
<i>Model 1 To Person ID</i>	<i>Model 2 To Person ID</i>
<i>Model 1 To Organization Name</i>	<i>Model 2 To Organization Name</i>
<i>Model 1 To Organization Type</i>	<i>Model 2 To Organization Type</i>
<i>Model 1 To Organization Person ID</i>	<i>Model 2 To Organization Person ID</i>
<i>Model 1 Interaction</i>	<i>Model 2 Interaction</i>
<i>Final Criteria</i>	<i>Final Criteria ID</i>
<i>Final Elimination</i>	<i>Final Interaction</i>
<i>Final Connect</i>	

### 2.2.2 DIE: Data collection record object

<i>Unique ID of record</i>	<i>Project name</i>
<i>Effective date</i>	<i>Label</i>
<i>Type (As-is, Should-be, etc.)</i>	<i>Subtype (Mail, In person, Telephone, etc.)</i>

<i>From person ID</i>	<i>From person name</i>
<i>To person ID</i>	<i>To person name</i>
<i>Frequency of interaction</i>	<i>Importance of interaction</i>
<i>20 Impact values</i>	<i>20 Frequency values</i>
<i>20 Duration values</i>	<i>Notes</i>
<i>Used?</i>	

*Used?* is a Boolean value employed in the process of creating data collection links (see 2.5.3 *Diagram 3: Creating data collection links*, page 39). It is initialized to "False."

### 2.2.3 *DIG*: Data collection link object

"Unconfirmed":

<i>Left person ID</i>	<i>Right person ID (= Left person)</i>
<i>Left person name</i>	<i>Right person name (= Left person)</i>
<i>20 impact values for left person</i>	<i>20 impact values for right person = undefined</i>
<i>Is left person confirmed? = True</i>	<i>Is right person confirmed? = False</i>

"Confirmed":

<i>Left person ID</i>	<i>Right person ID</i>
<i>Left person name</i>	<i>Right person name</i>
<i>20 impact values for left person</i>	<i>20 impact values for right person</i>
<i>Is left person confirmed? = True</i>	<i>Is right person confirmed? = True</i>

## 2.2.4 *ISR*: Issue record object

<i>ID</i>	<i>Label</i>
<i>Project</i>	<i>Description</i>
<i>Type</i>	<i>Name</i>
<i>Subtype</i>	

## 2.2.5 *ENE*: Emergent nomination element

<i>Person ID</i>	<i>20 nomination values</i> (initialized to 0)
<i>Person name</i>	<i>20 maximized impact values</i> (initialized to -1)

## 2.2.6 *ESE*: Emergent structure element

<i>Issue number</i>	<i>Master issue</i>
<i>Impact value</i> (Boolean)	<i>6 "slave" issues. numbered 5-0</i>

## 2.2.7 *CNC*: Connection object

<i>Parent record type</i>	<i>Child record type</i>
<i>Parent record ID</i>	<i>Child record ID</i>

In *CNC* objects, parent records are always issues (*ISR*), of type *master*, *slave*, or *meta*. Child records may be issues (*slave* or *meta*) or persons (*PRR*).

## 2.3 TOP-LEVEL PROGRAM STRUCTURE

The emergent structure analysis performs the following basic steps:

1. Select the data collection records to be included in an analysis.
2. Select the issues to include and create the issue tree.
3. Compute each person's position in the issue tree by counting the number of *nominations* that person received to each level.
4. Add the numbered *slave issues* to the database if they do not already exist.
5. Create the links between persons and slave issues.

To accomplish these steps, the program repeatedly processes data collection records from the database. At each successive stage, it transforms existing data objects into new objects having the characteristics needed for the next stage. The basic process is illustrated in figure 12.



Figure 12: Emergent structure objects

### 2.3.1 Program flow

Diagram 1 (page 19) illustrates the top-level steps of the emergence analysis data flow. Subsequent diagrams will describe individual steps in greater detail, as indicated. For detailed discussion of the numbered items in the diagram, see 2.3.2 *Diagram 1: Overall program structure*, page 20.

## EnCompass Emergent Structure Analysis

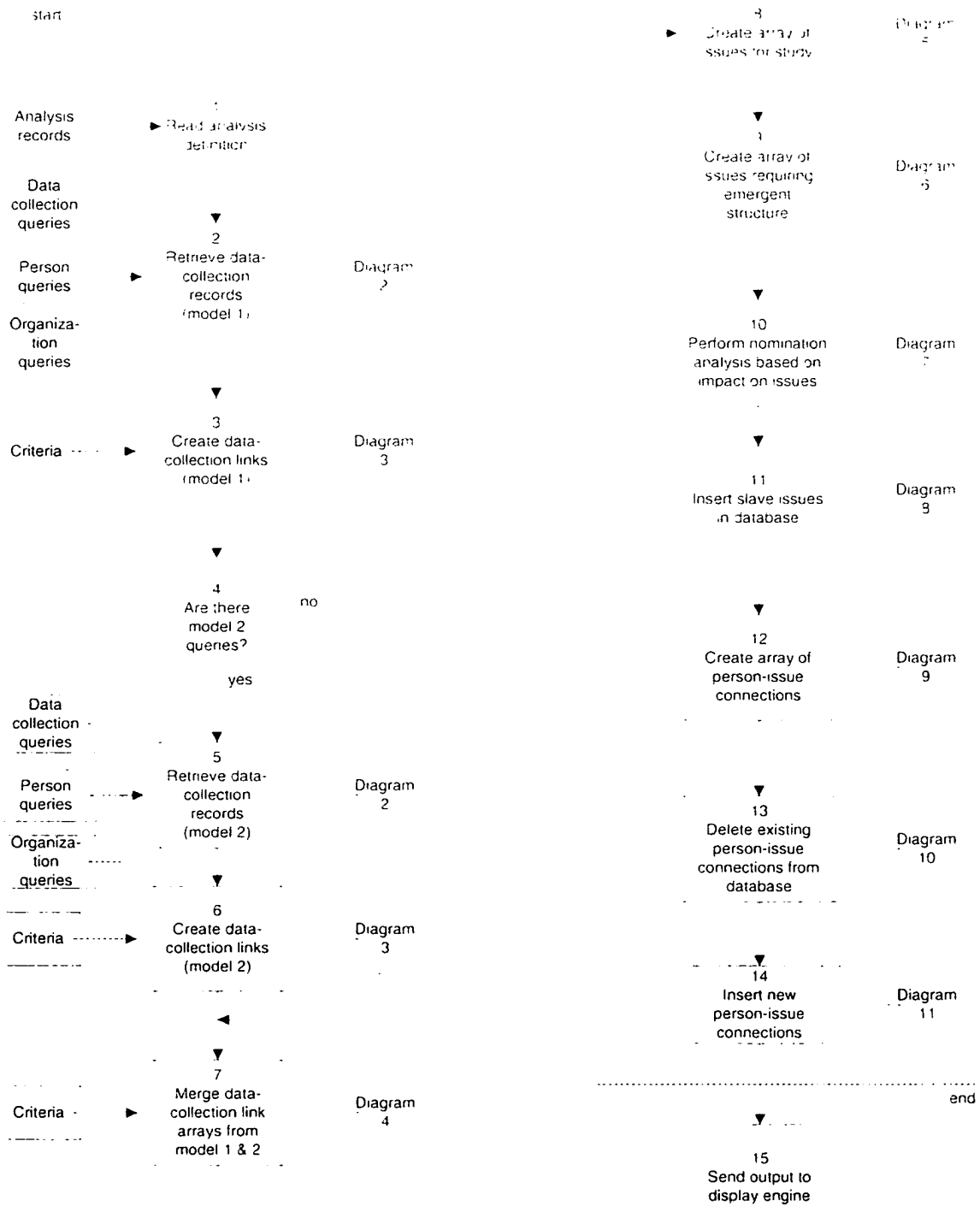


Diagram 1: Emergent Structure Analysis



### 2.3.2 Diagram 1: Overall program structure

The following notes apply to the numbered items in diagram 1. For explanations of *key concepts*, see page 11.

- 1 Read the *analysis definition* from the database as an object *ANR* (see page 14).  
An analysis comprises one or more *database queries*. These must include at least one *data collection* query, which defines desired ranges of values for overall interaction frequency, overall importance, and impact figures on one or more specific issues. In addition, the definition may also query the following database columns:

*From person*      *To person*

*From organization*   *To organization*

The analysis must also include an *Agreement Status* setting and an associated *Criteria* query. These specify ranges of impact values for one or more issues, and determine whether the query will search for records in which the *From Person's* and the *To Person's* assessments either *Agree*, *Disagree*, or *Either* (agree or disagree) within the range.

In the EnCompass application interface, the user selects from lists of existing named queries, stored in the associated database tables (*Data Collection*, *Person*, *Organization*, and *Criteria*). The model 1 selection panel appears as in figure 13. (The panel for the optional model 2 part of the analysis definition is identical.)

Figure 13: Model 1 query selection

For details, see 2.5.2 *Diagram 2: Retrieving data collection records* (page 37), item 2.1.

- 2 Construct an array of *data collection records (DIE)*, selecting records from the database which match the SQL queries in the model 1 analysis definition (diagram 2, page 23).
- 3 Match the array of records from step 2 against all other data collection records, transforming it into a new array of *data collection links* (class *DIG*). (Diagram 3, page 25.)
- 4 If there is a set of queries defined in the model 2 section of the analysis, process them the same as the model 1 queries. If there are no model 2 queries, proceed to 7.
- 5, 6 Duplicate steps 2 and 3 for the queries defined for model 2, if any. Create a separate *DIG* array for model 2.
- 7 Merge the *DIG* arrays from model 1 and model 2 into a single array, according to criteria specified in the analysis definition (see diagram 4, page 26). If there are no model 2 queries, this step simply transfers the model 1 *DIGs*.
- 8 Create an array of all the master issues in the current analysis (diagram 5, page 30).
- 9 Extract from the analysis definition those issues that require an emergent structure (diagram 6, page 31).
- 10 For all the *confirmed data collection links*, promote each node to the highest level for which it received the minimum number of *nominations* required by the analysis definition (diagram 7, page 32). This step transforms the *DIG* array into an array of *emergent nomination elements* (class *ENE*).
- 11 For the emergent structure analysis name specified in the analysis definition, insert a set of 6 *slave issues* into the database, if they are not already present (see diagram 8, page 33). These records duplicate the contents defined in the master issue, except that they are labeled with values from "Level 5" to "Level 0," corresponding to the possible impact values assigned by survey respondents.
- 12 From the results of the nominations calculated in item 10, create an array of *connection records* (class *CNC*). (See diagram 9, page 34.) These connections link persons to slave issues, in a child-parent relationship. The parent is the slave issue matching the highest level (0-5) to which a person was nominated.

- 13 Remove any existing person-issue connections from the database, if required (diagram 10, page 35).
- 14 The final step in the emergent structure analysis process is to insert the new array of issue-issue and person-issue connections in the database (diagram 11, page 36). The display engine which converts these connections to graphic form for presentation is an existing module outside the scope of this discussion (see 3.0 *Display options*, page 53).

## 2.4 PROGRAM FLOW DIAGRAMS

The following diagrams illustrate the individual steps in the emergent structure process. For context, see diagram 1 (page 19). For discussion, see 2.5 *Detailed discussion*, page 37.

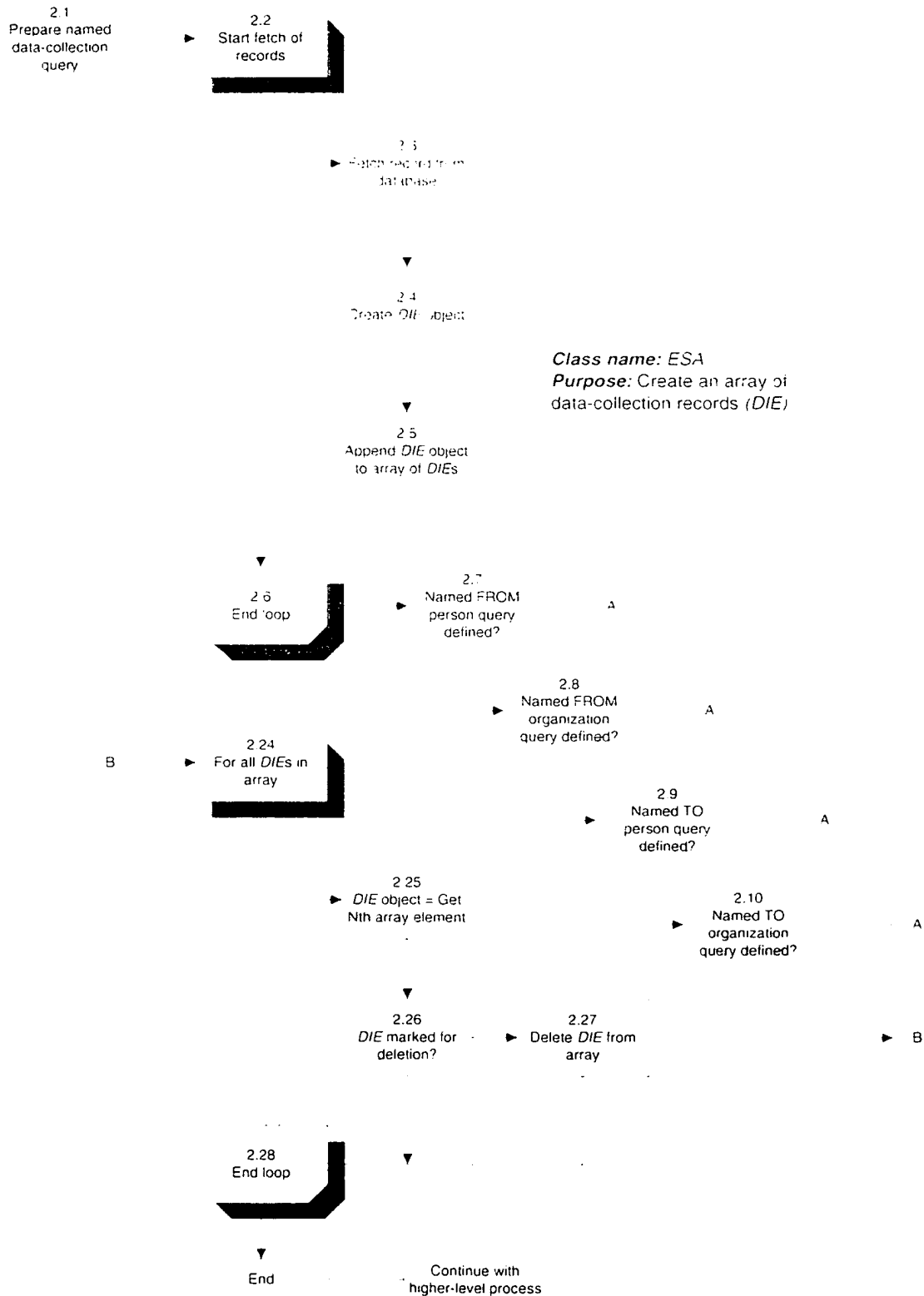


Diagram 2: Retrieving data collection records

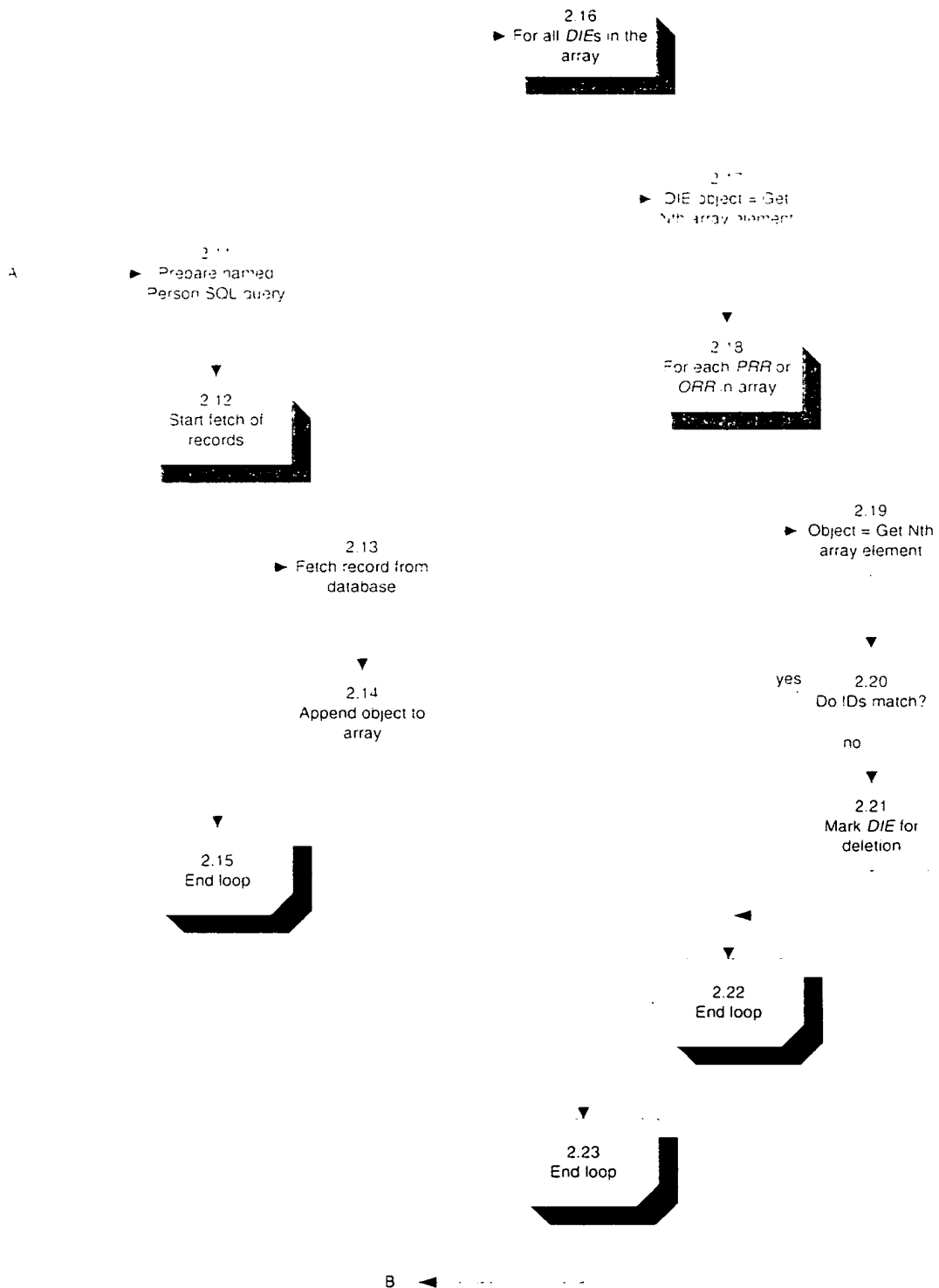


Diagram 2a: Creating data collection records, part 2

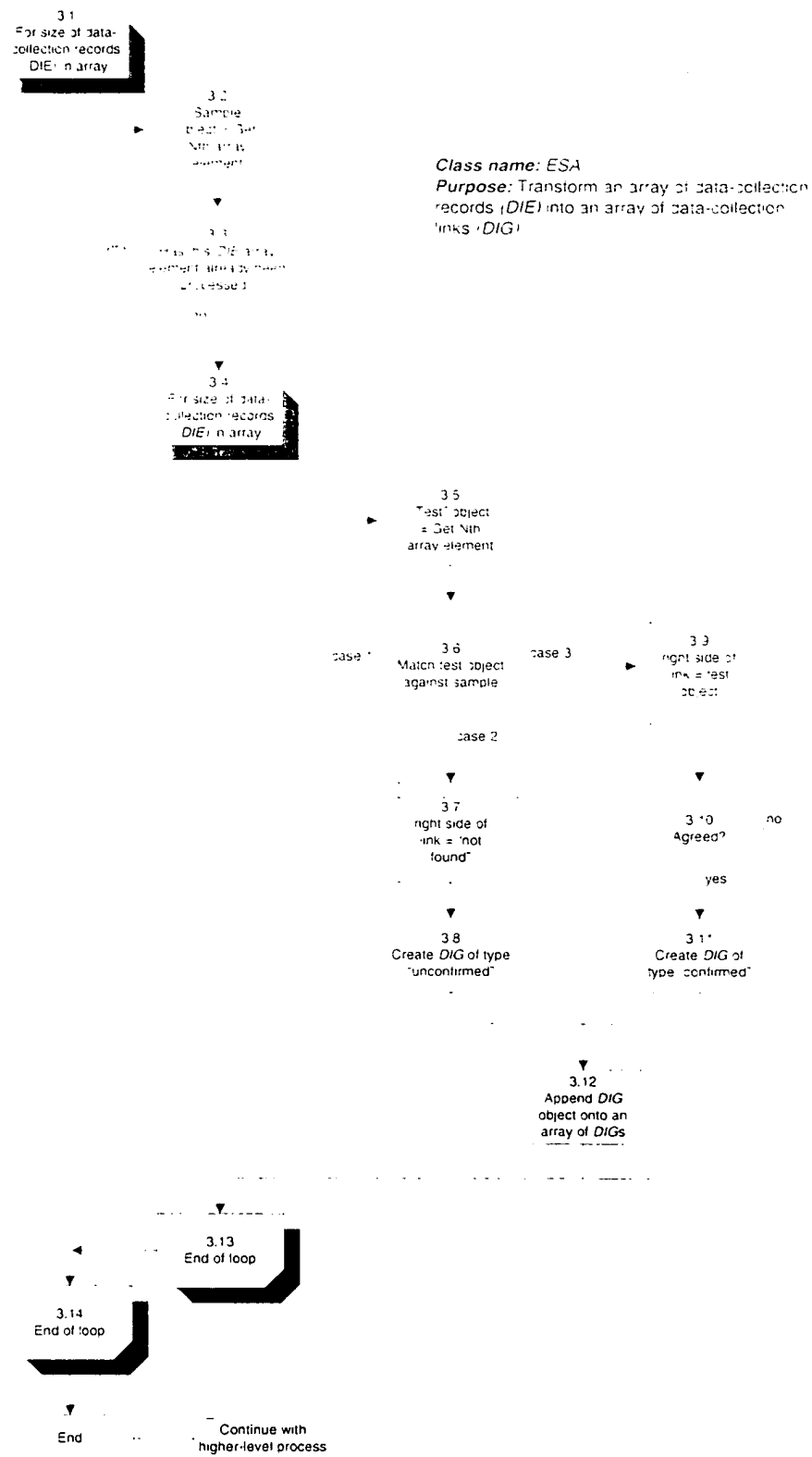


Diagram 3: Creating data collection links

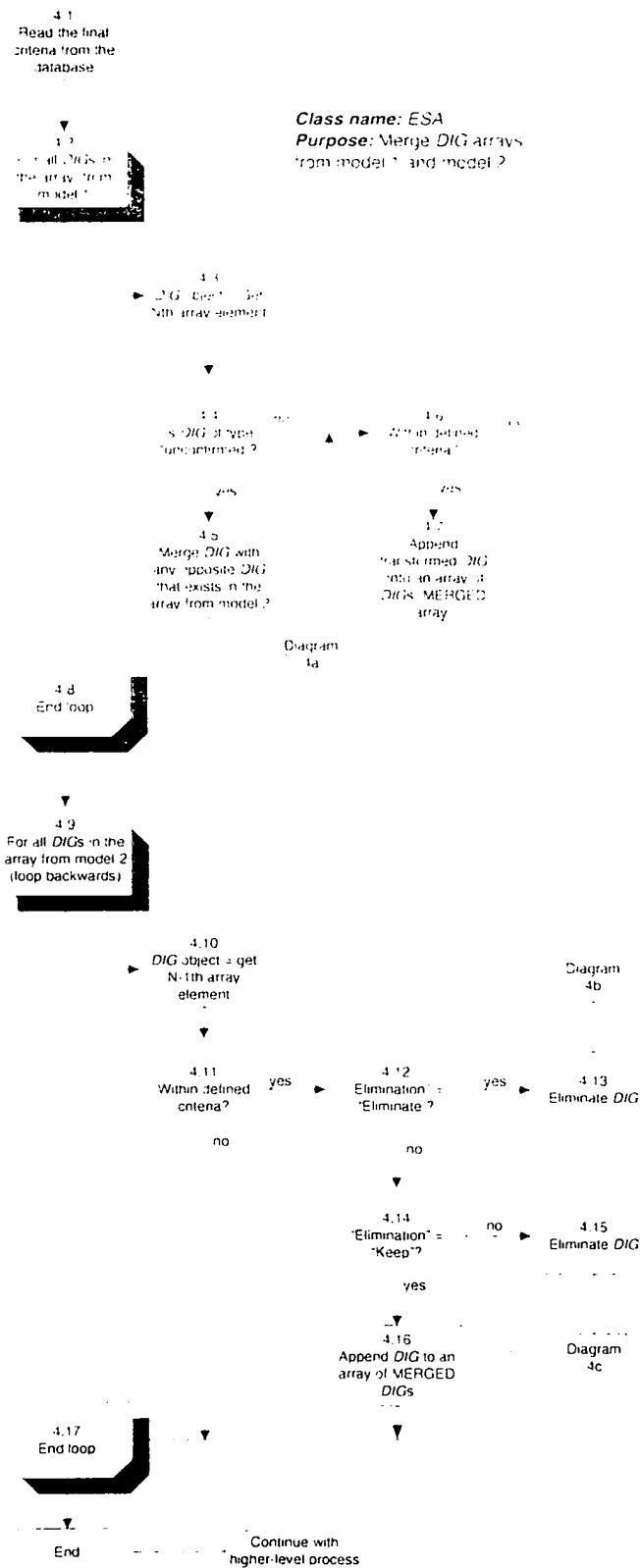


Diagram 4: Merging data collection models

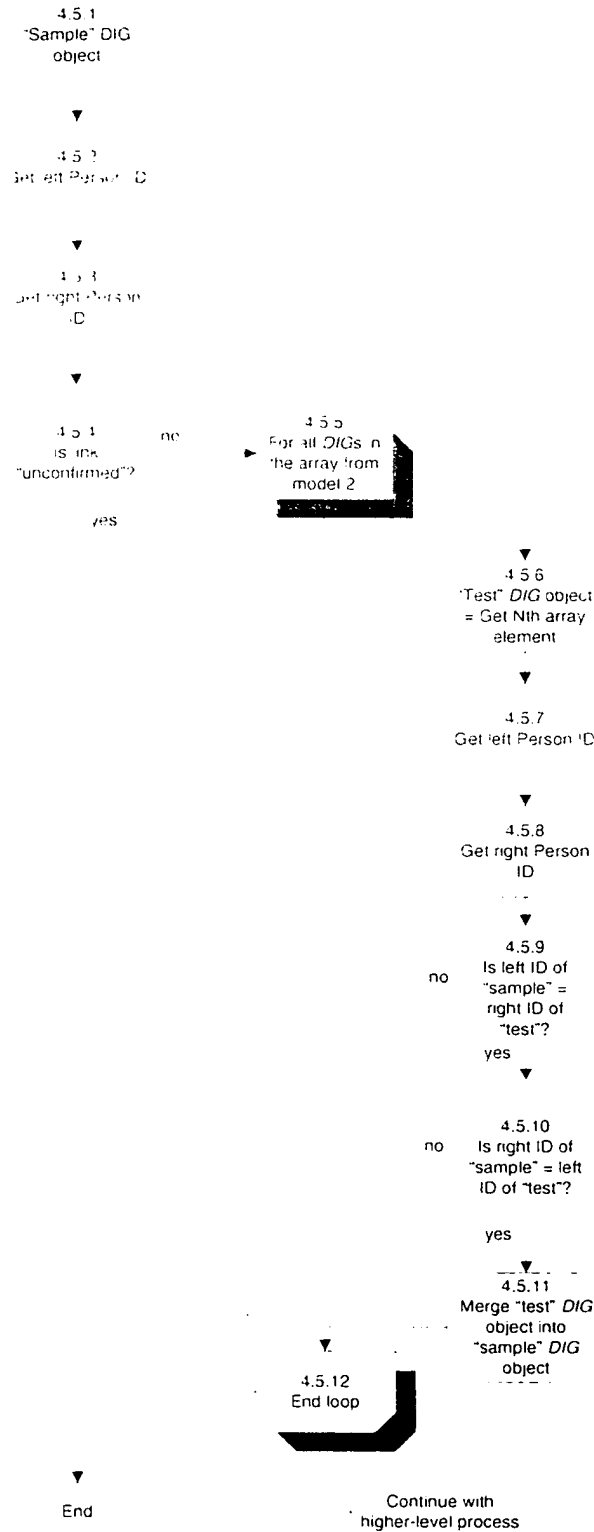


Diagram 4a: Merging model 2 DIGs



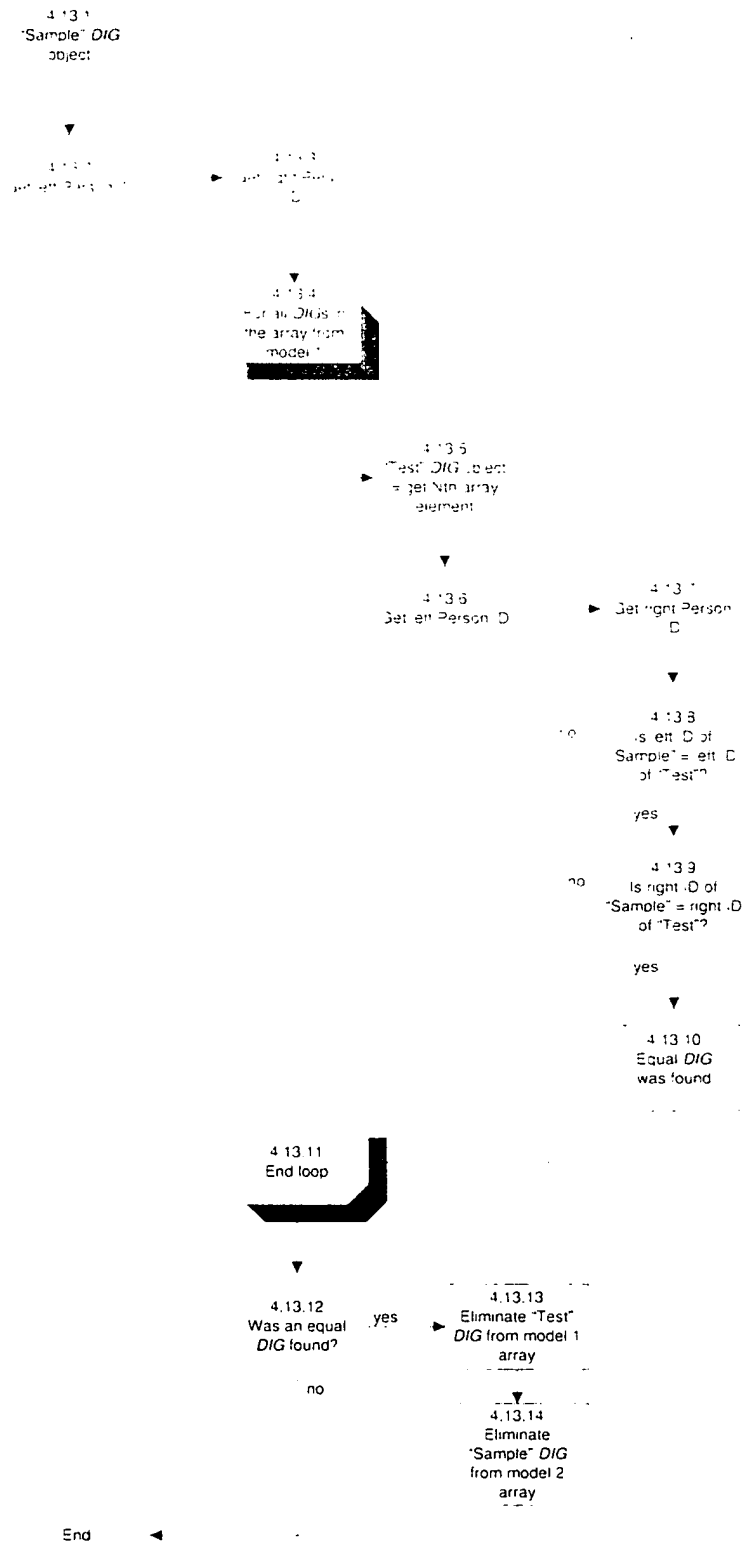


Diagram 4b: Eliminate if in both

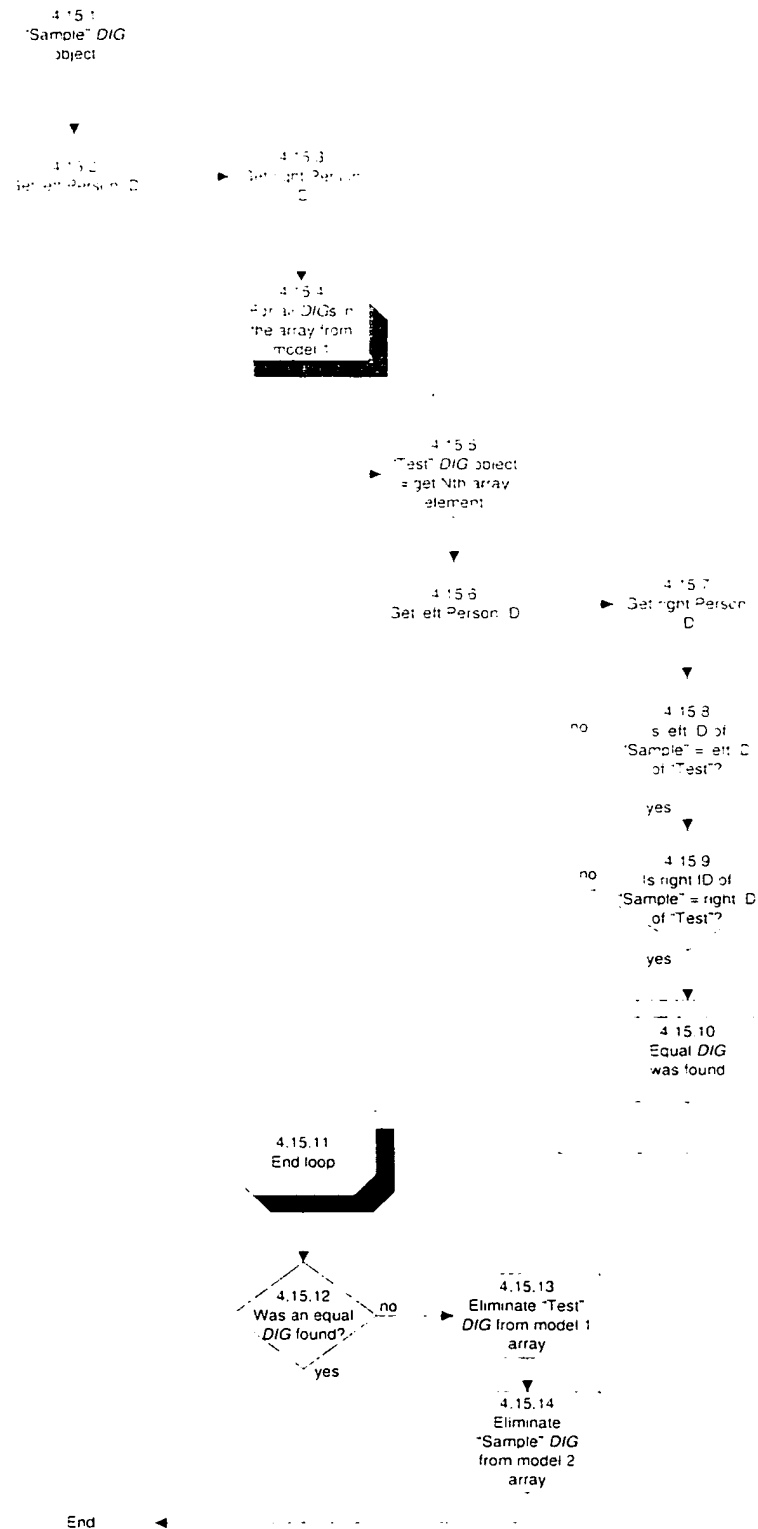


Diagram 4c: Keep if in both

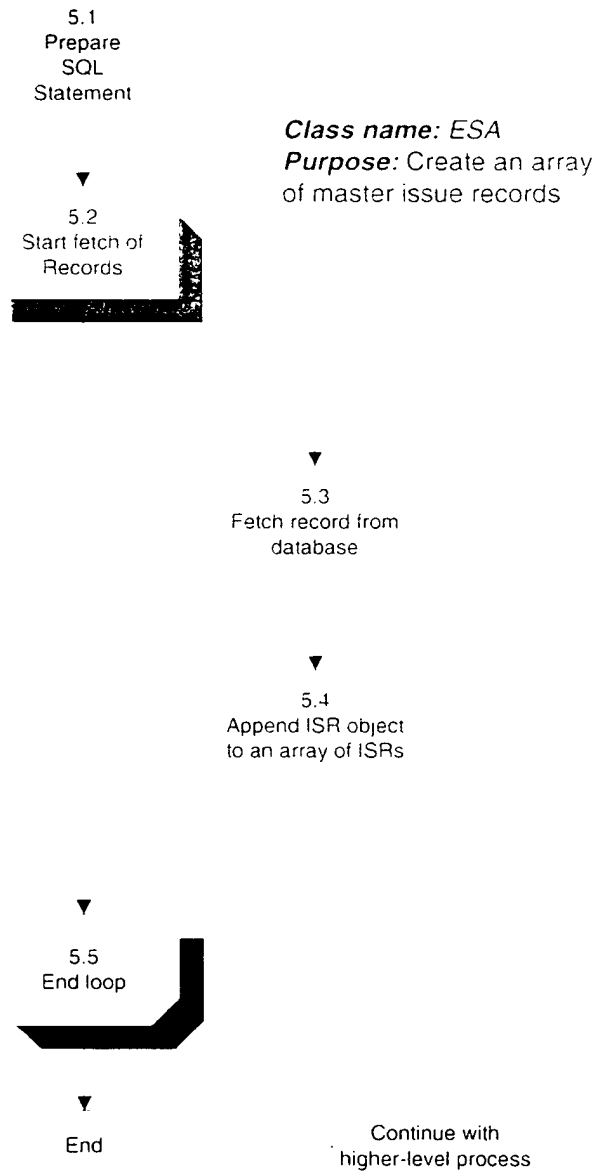


Diagram 5: Listing master issues

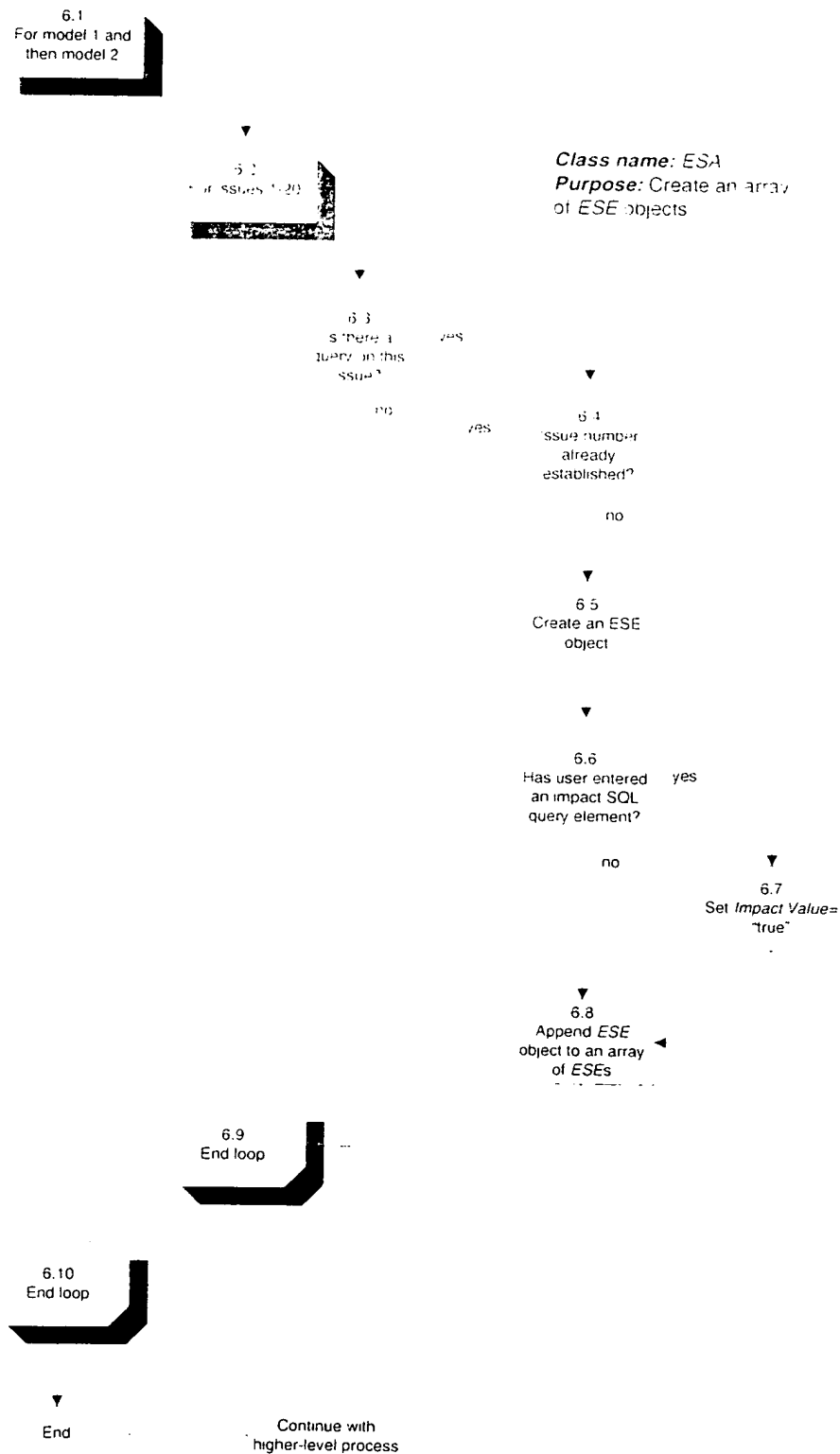


Diagram 6: Creating emergent structure elements

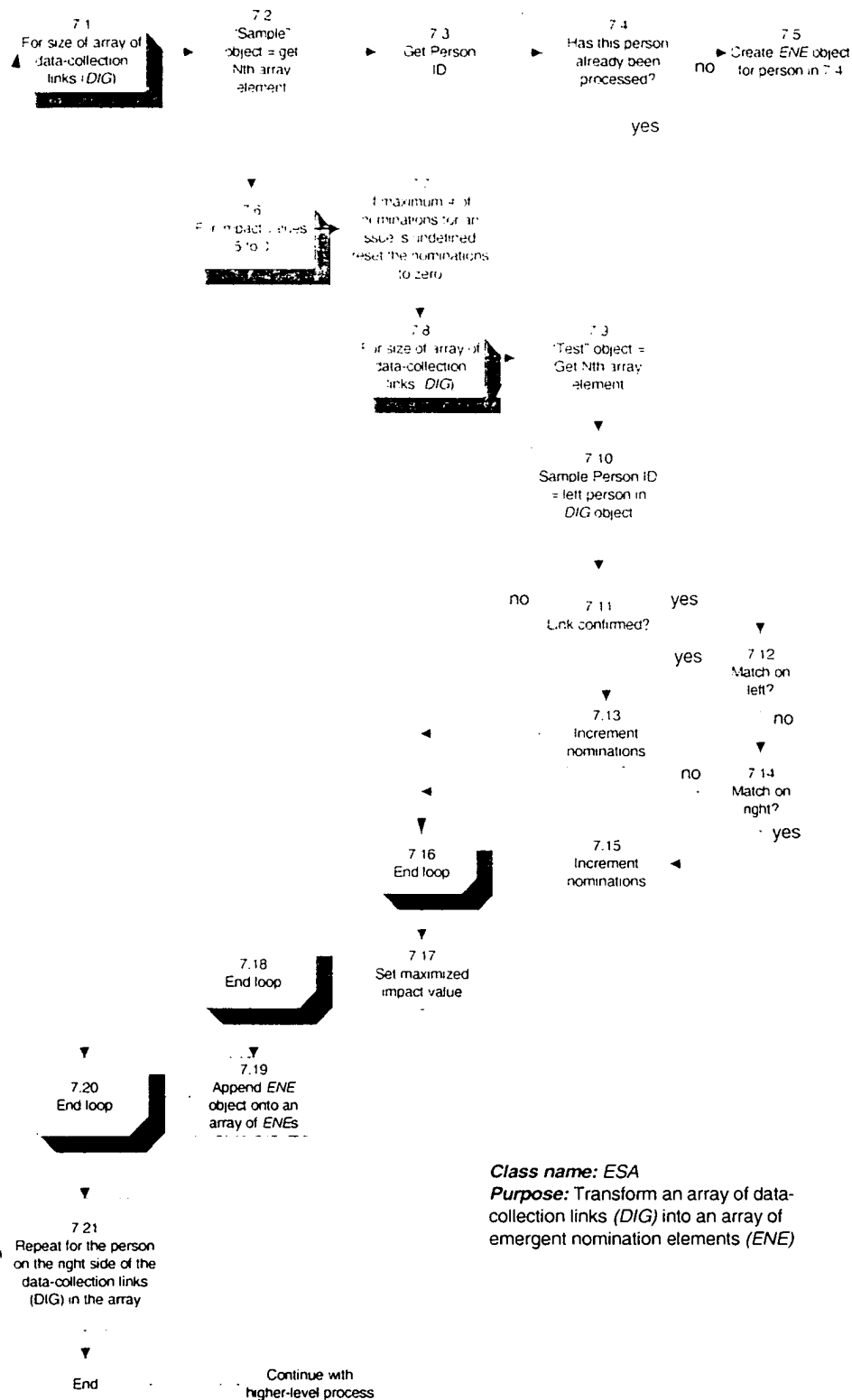


Diagram 7: Nomination

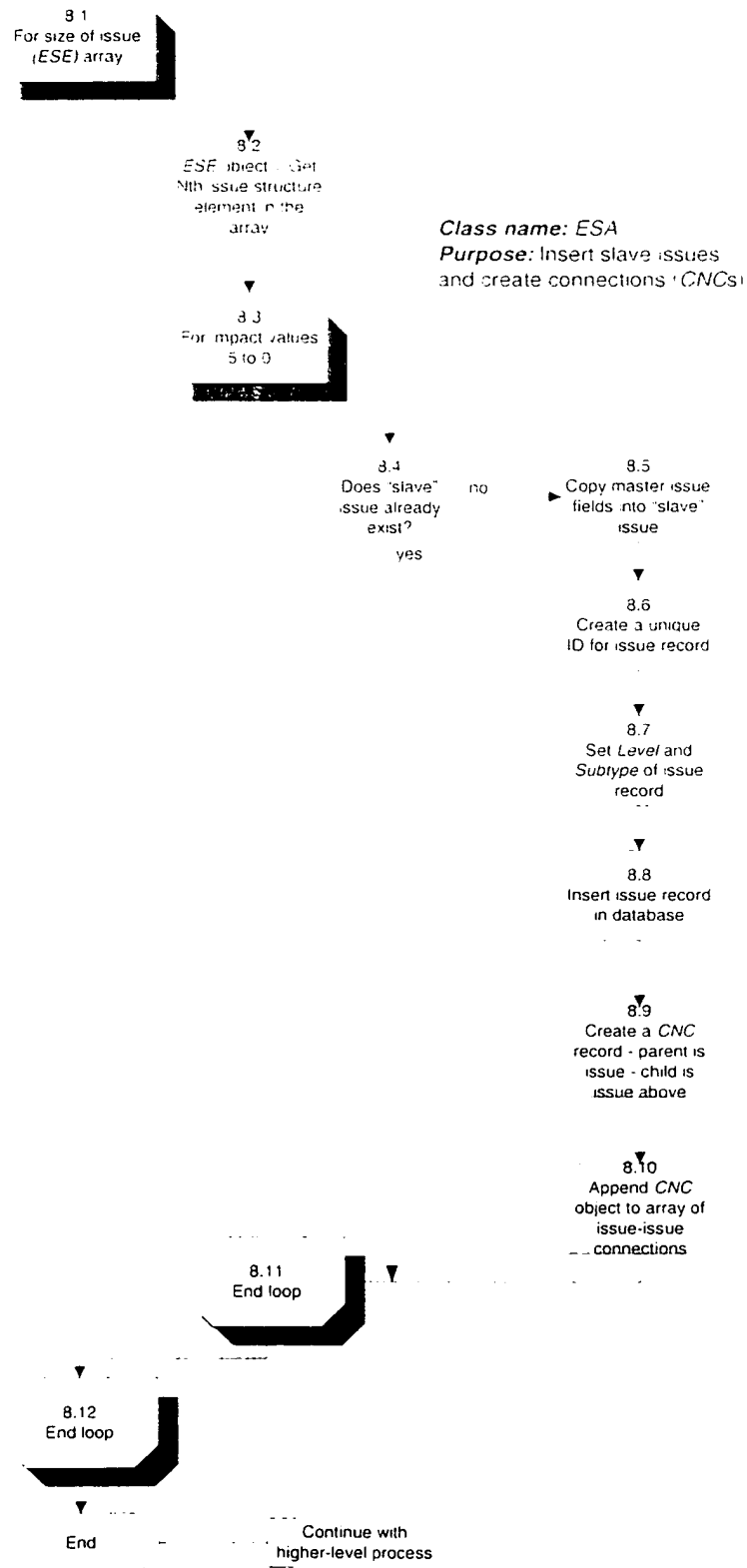


Diagram 8: Inserting slave issues

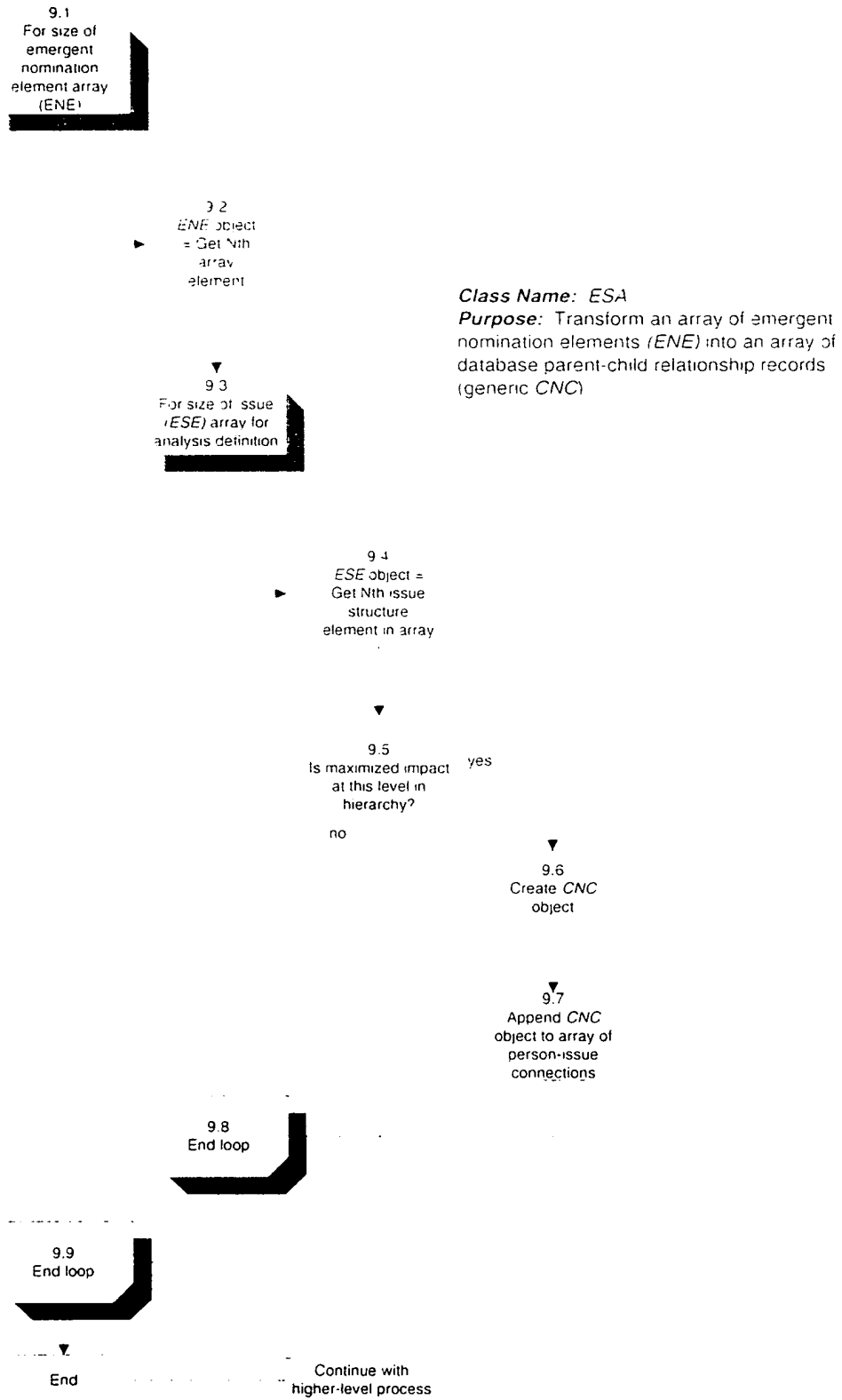


Diagram 9: Creating CNC objects

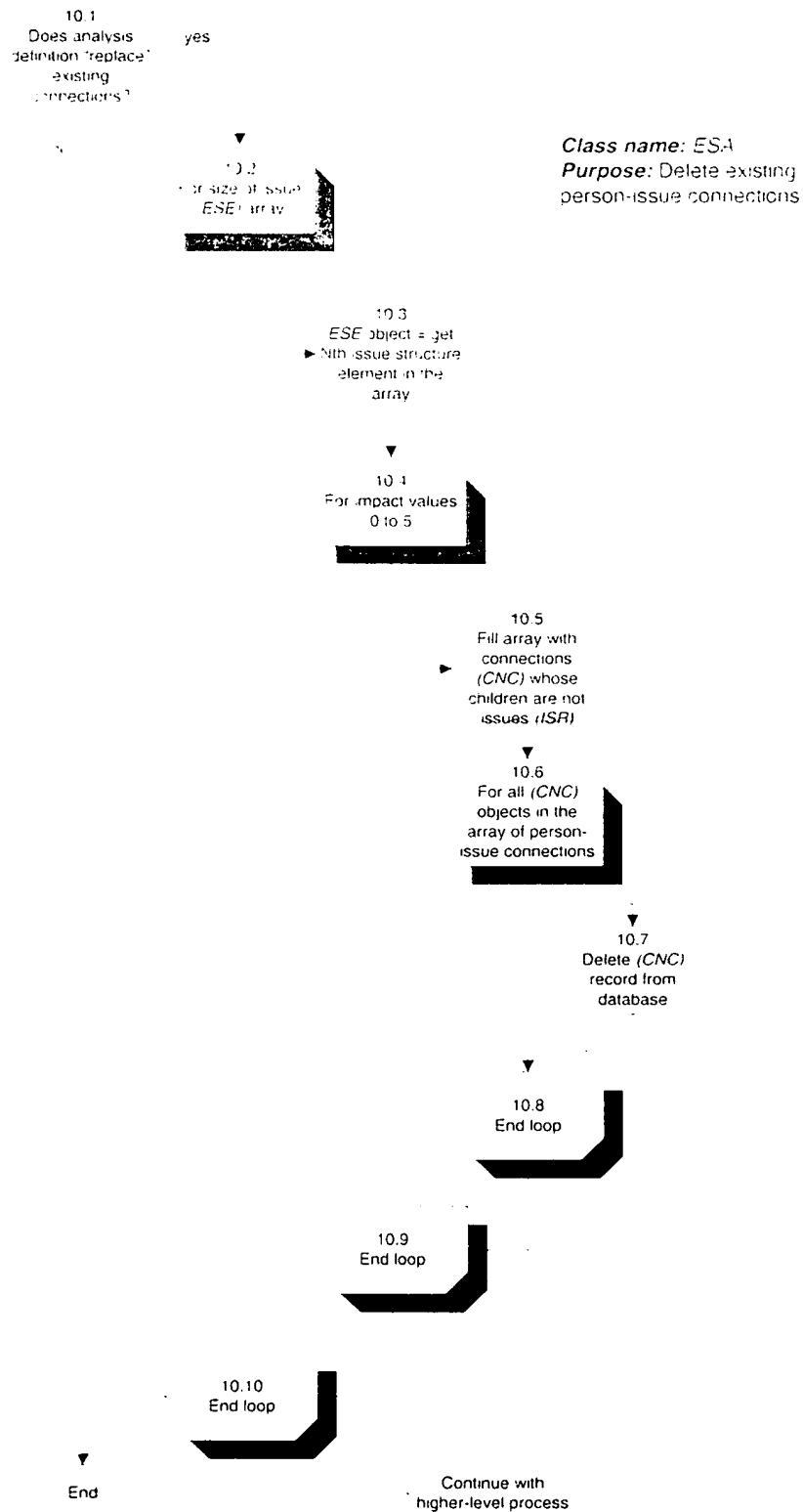


Diagram 10: Deleting existing connections



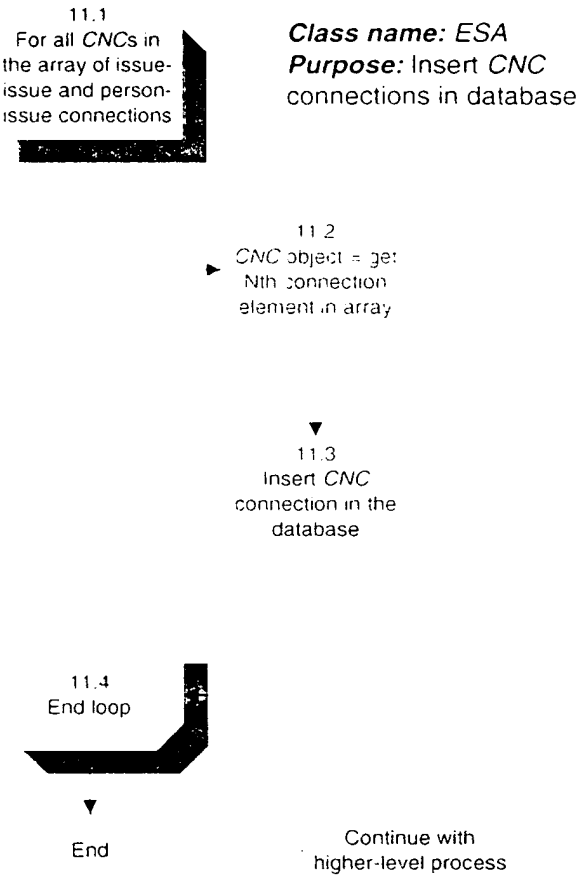


Diagram 11: Inserting new connections

## 2.5 DETAILED DISCUSSION

### 2.5.1 Introduction

Diagram 1 (page 19) and the associated discussion introduced the overall flow of the emergent structure analysis procedure. Diagrams 2-11 (pages 23-36) offered schematic views of the individual steps in the process. The following sections provide the annotations to diagrams 2-11. For definitions of *key concepts*, see page 11. For descriptions of data objects (e.g. *DIE*), see page 14.

### 2.5.2 Diagram 2: Retrieving data collection records

After reading the *analysis definition* to determine what *SQL queries* will be needed (diagram 1, item 1), the first step in constructing the emergent structure is to retrieve the matching records from the database (see diagram 2, page 23).

- 2.1 Parse the individual query conditions from the analysis definition and construct a SQL statement to retrieve the matching data collection records.

For example, an analysis might use a data collection query such as: "Frequency and Importance  $\geq 4$ ; Management Processes  $\geq 4$ ," thus searching for interactions with a high or very high overall frequency and importance, and a high or very high impact on the defined issue of "management processes" (internally, issue #6). If the current project is called "Sample," this query will generate the following SQL:

```
SELECT * FROM Data_Collection WHERE (ID LIKE '%') AND (PROJECT =  
'Sample') AND (PERSONFREQUENCY >= 4) AND (PERSONIMPORTANCE >= 4)  
AND (IMPACTVAL6 >= 4) ORDER BY ID
```

- 2.2. Loop: Perform steps 2.3-2.6 for every data collection record found.

- 2.3-2.4 Retrieve the next record and store it in memory as an object of class *DIE* (see page 16).

- 2.5 Append the *DIE* object to an array of *DIEs*.

- 2.6 Go to the next record.

- 2.7 Test whether the analysis definition includes an optional *From Person* query. If "true," go to step 2.11. If there is none, go to the next test (2.8).
- 2.8 Test whether the analysis definition includes an optional *From Organization* query. If "true," go to step 2.11. If there is none, go to the next test (2.9).
- 2.9 Test whether the analysis definition includes an optional *To Person* query. If "true," go to step 2.11. If there is none, go to the next test (2.10).
- 2.10 Test whether the analysis definition includes an optional *From Organization* query. If "true," go to step 2.11. If there is none, go to 2.24.

[Steps 2.11 through 2.23 form a procedure that may be repeated up to four times, once for each of the optional queries included in the analysis definition. The values used each time will depend on the type of query, as noted. (Refer to diagram 2a, page 24.)]

- 2.11 From the conditions in the analysis definition (model 1 or model 2), construct a SQL database query of the appropriate type. For example, an analysis might include a *From Person* query named "Executive," which searches for cases in which the person reporting the interaction is in the executive division. Again, if the project name is "Sample," this query generates the following SQL:

```
WHERE (ID LIKE '%') AND (PROJECT = 'Sample') AND (DIVISION = 'Executive')  
ORDER BY ID
```

*From Person* and *To Person* queries will retrieve records from the *Person* table;  
*From Organization* and *To Organization* queries will search the *Organization* table.

- 2.12 *Loop*: Perform steps 2.13-2.15 for every record in the appropriate table (*Person* or *Organization*).
- 2.13 Retrieve the next record as an object of type *PRR* (for person queries) or *ORR* (for organization queries).
- 2.14 Append the *PRR* or *ORR* to an array of *PRRs* or *ORRs*.

[Steps 2.16 through 2.23 compare the objects in the person (*PRR*) or organization (*ORR*) array just created against those in the *DIE* array, searching for matching record ID values. Nonmatching records are flagged for later deletion.]

- 2.16 *Loop 1*: Test every element in the *DIE* array as follows:

2.17 Get the next element in the array.

2.18 *Loop 2:* For every element in the (*PRR* or *ORR*) array from 2.12, do the following:

2.19 Get the next array element.

2.20 Test the *ID* values in the two objects according to the following scheme:

<u>for query type:</u>	<u>test <i>DIE</i> value:</u>	<u>against ID in:</u>
"From Person"	<i>From Person ID</i>	<i>PRR</i>
"From Organization"	<i>From Organization ID</i>	<i>ORR</i>
"To Person"	<i>To Person ID</i>	<i>PRR</i>
"To Organization"	<i>To Organization ID</i>	<i>ORR</i>

If the values match, retain the *DIE* object and go to the next element (2.22). If there is no match, mark the *DIE* object for deletion (2.21).

2.22-2.23 After testing all *PRR/ORR* records against every *DIE* for the current query type, check for the presence of the next type (2.7-2.10). When all types have been tested, go to 2.24.

(For steps 2.24 through 2.28, refer back to diagram 2.)

2.24 *Loop:* Perform the following steps for every element in the *DIE* array.

2.25 Get the next array element.

2.26 Check whether the object was marked for deletion in 2.21.

2.27 If it is marked, delete it from the array (2.28). If not, retain it and go to the next *DIE*.  
When all *DIEs* have been checked, return to the main procedure (diagram 1, item 3 or 6).

### 2.5.3 Diagram 3: Creating data collection links

After selecting *data collection records* from the database, according to queries specified in the *analysis definition*, the program transforms the array of data collection records into **data collection links** (class *DIG*). (See diagram 3, page 25.) A *DIG* object links an interaction reported by one person (the *left person*) with one reported by a second person (the *right person*). The *From person* and *To person* names on both sides of the

*DIG* link are compared end-for-end: if they match in both directions, the link is "confirmed," meaning that the right person also says that the interaction takes place (though the frequency, importance, and impact values may differ). If only in one direction, it is an "unconfirmed" interaction.

3.1 Begin with an array of *data collection records* (DIE) (diagram 1, item 2).

*Loop 1:* Perform the following steps on every element in the *DIE* array.

3.2 Get the next element in the array. Call this object "Sample." (Values for the left side of the *DIG* link will come from this object.)

3.3 Check the value of *Used?* If "true," this item has already been processed. Proceed to the next array element.

3.4 *Loop 2:* For each iteration of loop 1, perform the following steps on every element in the *DIE* array.

3.5 Get the next item in the array to compare against "Sample." Call this object "Test."

3.6 (a) Match the *From person ID* in "Sample" against the *To person ID* in "Test."

**Case 1:** If the match fails, discard "Test" (go to 3.13).

If the match is true, (b) match the *From person ID* in "Test" against the *To person ID* in "Sample."

**Case 2:** If match (b) fails, the link is *unconfirmed*. Go to 3.7.

**Case 3:** If match (b) is true, the link is *confirmed*. Go to 3.9.

3.7 Store the fact that there is no match for the right side of the link.

3.8 Create an "unconfirmed" data collection link object (*DIG*) (see page 16). Populate the *Right person ID* and *Right person name* with dummy values, copied from "Sample." The *Right person confirmed* value is unused here and will be discarded later. Go to 3.12.

3.9 Store the values from the test object. This will be the right side of the data collection link (*DIG*).

3.10 Compare the issue-impact values in the sample and test objects. If they fall within the agreement criteria in the analysis definition, go to 3.11. If not, exit loop 2 (go to 3.13).

- 3.11 Create a "confirmed" data collection link object (*DIG*) (see page 16). The *Right person confirmed value* is unused here and will be discarded later.
- 3.12 Append the new *DIG* object ("confirmed" or "unconfirmed") to the array being built.
- 3.13 Go to the next test object.
- 3.14 Go to the next sample object. When all data collection records in the *DIE* array have been examined, return to the main procedure (diagram 1, item 4).

## 2.5.4 Diagram 4: Merging data collection models

Every analysis must include at least one set of queries (model 1) to retrieve data collection records from the database; the program stores the results in an array of data collection links (*DIG*). Optionally, an analysis may also include a second set of queries for *model 2*, the results of which are stored in a second *DIG* array. In this case, the results of the two query sets must be merged into a single *DIG* array for further processing (diagram 4, page 26).

- 4.1 The parameters for merging the model 1 and model 2 arrays are set by the user, in the *Show Results* panel in the program interface (figure 14). The settings are stored in the database for reuse.

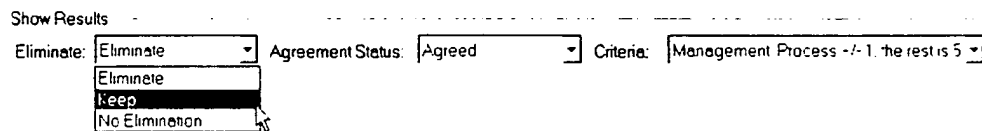


Figure 14: *Show Results* panel

The *Eliminate* setting determines what happens to records that appear in both models, by specifying the action to take when two compared *DIG* objects contain the same pair of left and right person IDs (in either order and regardless of impact values). Options are *eliminate*, *keep*, and *no elimination*.

- "Eliminate" discards both records (binary subtraction)
- "Keep" retains both records (binary addition)
- "No elimination" performs no checks and retains any record that appears in either model

*Agreement Status* may be *agreed*, *disagreed*, or *either*.

- "Agreed" retains the records if their impact values match, within the range specified by the Criteria setting.
- "Disagreed" retains the records if their impact values do NOT match, within the range specified by the Criteria setting.
- "Either" performs no checking of issue-impact values.

*Criteria* specify ranges of issue-impact values for the agreement status condition.

- 4.2 *Loop*: Perform steps 4.3-4.7 for every element in the array of *DIGs* from model 1. This loop will move the model 1 *DIGs* into the new merged *DIG* array.
- 4.3 Get the next array element as a new *DIG* object.
- 4.4 The object was created as either type "confirmed" or "unconfirmed" (diagram 3, steps 3.8, 3.11). If it is "unconfirmed," go to 4.5. If not, go to 4.6.
- 4.5 Check the *DIG* object against the array from model 2 to see whether a match for the right side of this link exists there. If it does, merge the *DIGs* to transform this into a "confirmed" link (see 2.5.4.1 *Diagram 4a: Merging model 2 DIGs*, page 43).
- 4.6 Check the impact values in the *DIG* object against the ranges specified in the "Criteria" query and the "Agreement Status" condition (4.1). If they do not match, go to the next object (4.8). If they match, go to 4.7.
- 4.7 Append the *DIG* (now of type "confirmed") to a new array of *merged DIGs*.
- 4.8 Get the next *DIG* object from the model 1 array. When all model 1 *DIGs* have been examined, continue to 4.9.
- 4.9 *Loop*: Perform the remaining steps for every element in the array of *DIGs* from model 2. (Loop backward from the end of the array, in order to avoid reindexing when elements are deleted.) This loop will merge the model 2 *DIGs* into the new merged array.
- 4.10 Get the next element in the array as an object of type *DIG*.
- 4.11 Check the impact values in the *DIG* object against the ranges specified in the "Criteria" query and the "Agreement Status" condition (4.1). If they do not match, get the next object (4.17). If they match, go to 4.12.

- 4.12 Check the "Eliminate" parameter (4.1). If it is "Eliminate," go to 4.13. If not, go to 4.14.
- 4.13 See 2.5.4.2 *Diagram 4b: Eliminate if in both*, page 44.
- 4.14 Check the "Eliminate" parameter (4.1). If it is "Keep," go to 4.15. If not, go to 4.16.
- 4.15 See 2.5.4.3 *Diagram 4c: Keep if in both*, page 45.
- 4.16 The *DIG* object from model 2 meets all the criteria for merging. Add it to the *merged DIG* array.
- 4.17 Get the next element in the model 2 array. When finished, return to the main procedure (diagram 1, item 8).

#### **2.5.4.1 Diagram 4a: Merging model 2 DIGs**

The type "unconfirmed" indicates that the *From Person* and *To Person* values on the left side of a data collection link (*DIG*) do not match those on the right side. It is always possible, however, that a match will exist in a data collection record returned by a model 2 query. In this case, the link must be added to the array of merged *DIGs*. (Refer to diagram 4a, page 27.)

- 4.5.1 Call the *DIG* object "Sample."
- 4.5.2 Read the value of *Left person ID* from "Sample."
- 4.5.3 Read the value of *Right person ID* from "Sample."
- 4.5.4 The object was created as either type "confirmed" or "unconfirmed" (diagram 3, steps 3.8, 3.11). If it is "unconfirmed," return to the main procedure (4.6). If not, go to 4.5.5.
- 4.5.5 *Loop*: Test "Sample" against every *DIG* in the array from model 2.
- 4.5.6 Get the next object in the array. Call it "Test."
- 4.5.7 Read the value of *Left person ID* from "Test."
- 4.5.8 Read the value of *Right person ID* from "Test."
- 4.5.9 Match the *Left person ID* in "Sample" against the *Right Person ID* in "Test." If the match is false, go to the next Test object (4.5.12). If true, go to 4.5.10.



4.5.10 Match the *Right person ID* in "Sample" against the *Left Person ID* in "Test." If the match is false, go to the next Test object (4.5.12). If true, go to 4.5.11.

4.5.11 Combine "Sample" and "Test" into a single "confirmed" *DIG* object.

4.5.12 Get the next "Test" object from the model 2 array. When all *DIGs* have been tested, return to the main procedure (4.6)

#### **2.5.4.2 Diagram 4b: Eliminate if in both**

When the *Eliminate* setting is "Eliminate" (4.1), any *DIG* object found in both the model 1 and model 2 arrays is discarded. (Refer to diagram 4b, page 28.)

4.13.1 Get the *DIG* object from the model 2 array. Call it "Sample."

4.13.2 Read the value of *Left Person ID* from "Sample."

4.13.3 Read the value of *Right Person ID* from "Sample."

4.13.4 *Loop*: Perform steps 4.13.5-4.13.10 for every element in the array of *DIGs* from model 1.

4.13.5 Get the next *DIG* object in the array. Call it "Test."

4.13.6 Read the value of *Left Person ID* from "Test."

4.13.7 Read the value of *Right Person ID* from "Test."

4.13.8 Compare the *Left Person ID* in "Sample" against the *Left Person ID* in "Test." If they match, go to 4.13.9. If not, go to the next Test object.

4.13.9 Compare the *Right Person ID* in "Sample" against the *Right Person ID* in "Test." If they match, go to 4.13.10. If not, go to the next Test object.

4.13.10 Record the fact that the model 1 and model 2 *DIGs* match.

4.13.11 Get the next *DIG* in the model 1 array. When all *DIGs* have been tested, go to 4.13.12

4.13.12 Check the result of step 4.13.10 to see whether an equal *DIG* was found. If no, exit. If yes, go to 4.13.13.

4.13.13 Delete the *DIG* object "Test" from the model 1 array.

4.13.14 Delete the *DIG* object "Sample" from the model 2 array. Return to 4.17.

### 2.5.4.3 Diagram 4c: Keep if in both

When the *Eliminate* setting is "Keep" (4.1), any *DIG* object found in both the model 1 and model 2 arrays is retained. (Refer to diagram 4c, page 29.)

- 4.15.1 Get the *DIG* object from the model 2 array. Call it "Sample."
- 4.15.2 Read the value of *Left Person ID* from "Sample."
- 4.15.3 Read the value of *Right Person ID* from "Sample."
- 4.15.4 *Loop*: Perform steps 4.15.5-4.15.10 for every element in the array of *DIGs* from model 1.
- 4.15.5 Get the next *DIG* object in the array. Call it "Test."
- 4.15.6 Read the value of *Left Person ID* from "Test."
- 4.15.7 Read the value of *Right Person ID* from "Test."
- 4.15.8 Compare the *Left Person ID* in "Sample" against the *Left Person ID* in "Test." If they match, go to 4.15.9. If not, go to the next Test object.
- 4.15.9 Compare the *Right Person ID* in "Sample" against the *Right Person ID* in "Test." If they match, go to 4.15.10. If not, go to the next Test object.
- 4.15.10 Record the fact that the model 1 and model 2 *DIGs* match.
- 4.15.11 Get the next *DIG* in the model 1 array. When all *DIGs* have been tested, go to 4.15.12
- 4.15.12 Check the result of step 4.15.10 to see whether an equal *DIG* was found. If yes, exit. If no, go to 4.15.13.
- 4.15.13 Delete the *DIG* object "Test" from the model 1 array.
- 4.15.14 Delete the *DIG* object "Sample" from the model 2 array. Return to 4.17.

### 2.5.5 Diagram 5: Listing master issues

This step creates an array of all the *master issues* in the current project. (See diagram 5, page 30.)

- 5.1 Prepare a SQL statement to retrieve records from the database's *Issue* table. The SQL query has the form `SELECT * FROM ISSUE WHERE Project = "Project name" AND Subtype = "Master issue."`
- 5.2 *Loop*: Perform the following steps for each issue record in the database.
- 5.3 Fetch the record from the database. (The SQL query assures that only issues of subtype "master" are retrieved.) Create an "issue record" object (*ISR*) in memory (see page 17).
- 5.4 Append the new object to an array of master issues.
- 5.5 Go to the next issue record.

### 2.5.6 Diagram 6: Creating emergent structure elements

This step (diagram 6, page 31) creates an array of *emergent structure elements* (class *ESE*; see page 17), which will be used when creating the person-issue connections.

- 6.1 *Loop*: Perform the following steps first for the model 1 analysis, then repeat for model 2, if any. (In practice, the EnCompass application allows for two analysis models; however, the process itself can accommodate any number of models within a single analysis.)
- 6.2 *Loop*: Perform the following steps for issues 1 through 20. (Most projects do not use all 20 issues; undefined issues will simply follow the "no" branch in step 6.3.)
- 6.3 Check whether any of the SQL queries defined in the analysis include this issue. If not, go to the next issue (6.9).
- 6.4 Prevents duplication: an issue number may not appear twice.
- 6.5 Create an *ESE* object. *Impact value* is initialized to "False."
- 6.6 Check whether the SQL query specifies an impact value for this issue. If no, go to 6.8.
- 6.7 A SQL query in the analysis specifies an impact value for this issue. Change the object's *impact value* setting to "True."
- 6.8 Add the *ESE* object to the array being built.
- 6.9 Go to the next issue.

6.10 Go to the next model in the analysis definition.

## 2.5.7 Diagram 7: Nomination

Once we have created the data collection links and listed the issues to be included in the analysis, the next step is to assign each person node to its appropriate level in the structure, as determined by the number of *nominations* to each level that each person received from other survey respondents. The nomination process transforms an array of *data collection links (DIG)* into an array of new objects called *emergent nomination elements (ENE)*. (See diagram 7, page 32.)

The input data for this step is an array of data collection links (*DIG*). Referring to diagram 1 (page 19), this will be either the single array from step 3 or the merged array from step 7.

- 7.1 *Loop 1:* Perform the following steps for each data collection link (*DIG*) in the array.
- 7.2 Read the next element in the *DIG* array. Call this object "Sample."
- 7.3 Get a *Person ID* value from "Sample." On the first pass (see 7.21), get the *Left person ID*. On the second pass, get the *Right person ID*.
- 7.4 If an *ENE* object already exists for this person ID, exit the loop (go to 7.20).
- 7.5 For the person in 7.3, create a new *emergent nomination element* (class *ENE*) (see page 17). The *maximized impact value* will record the highest impact value assigned to this person.
- 7.6 *Loop 2:* Perform the following steps for every impact value from 5 to 0. On each iteration, the loop counter corresponds to the number of the "slave" issue being tested. This value will be used in 7.12 and 7.14.
- 7.8 *Loop 3:* Compare each element in the array with every other element.
- 7.9 Get the next element in the *DIG* array; call this object "Test."
- 7.10 Store a *Person ID* from "Test." On the first pass (see 7.21), get the *Left person ID*. On the second pass, get the *Right person ID*.
- 7.11 Read the value of *Is left person confirmed?* from "Sample." If "false," go to the next test object (7.8). If "true," go to 7.12.

- 7.12 Test the *Left person ID* in "Sample" against the *Left person ID* in "Test." If they match, go to 7.13. If not, go to 7.14.
- 7.13 For each issue value, if the impact value in "Test" is greater than the value of the loop counter (i.e., the issue currently being examined) and *Maximized impact value* has not been set, then increment the number of nominations by one.
- On the first pass (see 7.21), this step operates on the right side of the *DIG* object.  
On the second pass, it operates on the left side.
- 7.14 Test the *Left person ID* in "Sample" against the *Right person ID* in "Test." If they match, go to 7.15. If not, go to 7.16.
- 7.15 For each issue, if the impact value in "Test" is greater than the value of the loop counter (i.e., the issue currently being examined) and *Maximized impact value* has not been set, then increment the number of nominations by one.
- On the first pass (see 7.21), this step operates on the left side of the *DIG* object.  
On the second pass, it operates on the right side.
- 7.17 For each issue, if the number of nominations is greater than or equal to the number of nominations set in the analysis AND *maximized impact value* has not been set, then set the *maximized impact value* of the *ENE* object equal to the issue value.
- 7.19 Add the new *ENE* object to the array being built.
- 7.21 Return to 7.1 and repeat for right side of the *DIG* array element.

### 2.5.8 Diagram 8: Inserting slave issues

In order to provide a structure to which the person-issue records can be connected, the program creates a set of *slave issues* for each *master issue* and adds them to the database (diagram 8, page 33). These records duplicate the contents defined in the master issues, except that they are labeled with values from "Level 5" to "Level 0," corresponding to the possible issue-impact values assigned by survey respondents. The database records are linked hierarchically: "Level 5" is a child of "Master," "Level 4" of "Level 5," and so on. Figure 15 shows the basic connection structure of a single master and 6 slave issues.

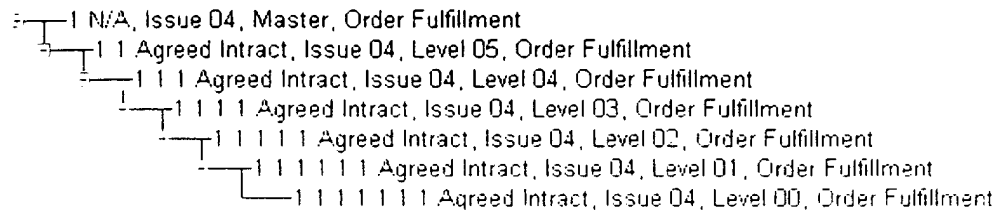


Figure 15: Master and slave issue tree

This procedure takes as its input the array of emergent structure elements (*ESEs*) created in diagram 1, step 9.

- 8.1 *Loop*: Perform the following steps for every element in the array of *ESE* objects.
- 8.2 Get the next element in the *ESE* array.
- 8.3 *Loop*: Perform steps 8.4-8.10 for every impact value from 5 to 0.
- 8.4 Check whether the slave issue already exists.
- 8.5 Copy the data from the master issue into the slave issue. The slave issue is one of six objects of type *ISR*, contained within the *ESE* object.
- 8.6 Assign a unique database ID number to the slave issue record.
- 8.7 Set values of *level* and *subtype* for the slave issue. *Level* is an integer from 5 to 0, corresponding to the current loop counter value. *Subtype* is a text string from "Level 5" to "Level 0."
- 8.8 Insert the new slave issue in the database.
- 8.9 Create a connection record (class *CNC*; see page 17). The child record is the *ISR* object created in 8.6; the parent is the *ISR* at one level above it.
- 8.10 Append the *CNC* object to an array of *CNC* objects.
- 8.11 Repeat for every issue-impact value from 5 to 0.
- 8.12 Get the next element in the array of *ESEs*. When all emergent structure elements have been examined, return to the main procedure (diagram 1, item 12).

## 2.5.9 Diagram 9: Creating connection records

The abstract nomination elements created in step 10 (diagram 7) must now be transformed into connection records (class *CNC*). Each *CNC* object connects a single person to the *slave issue* representing the highest level to which the person was nominated for a given issue (see diagram 9, page 34).

- 9.1 *Loop 1:* Perform the following steps for every element in the *ENE* array.
- 9.2 Get the next element in the array.
- 9.3 *Loop 2:* Perform the following steps for each of the issues included in the *analysis definition* (diagram 1, item 8).
- 9.4 Get the next issue structure object (*ESE*) (see page 17).
- 9.5 Read the *Impact value* from the *ESE* object. If it matches the *Maximized impact value* in the *ENE* object, go to 9.6.
- 9.6 Create a *CNC* object (see page 17). The parent record is the *slave issue* at the appropriate level; the child is the person. Each *CNC* link, therefore, connects a person to a slave issue at one of the levels 5 through 0.
- 9.7 Append the *CNC* object to an array of *CNC* objects. This is the same array that was created in step 8.10; in this case, the child record is a person (*PRR*).
- 9.8 Get the next *ESE* object in the array.
- 9.9 Get the next *ENE* object. When all nomination elements have been transformed, return to the main procedure (diagram 1, item 13).

The *CNC* links are the output from the emergent structure analysis. After the array of links is inserted in the database (diagram 1, item 14), they are available for processing by the display engine (see 3.0 *Display options*, page 53).

## 2.5.10 Diagram 10: Deleting existing connections

Connection Options

☒ Replace
 ☐ Append

Issue Name:

Nominations:

Figure 16: Replace existing connections

The *replace* setting removes from the database any emergent structure created by previous analyses. *Append* leaves existing structures in place, allowing the analyst to combine the results of multiple analyses in a single display. (Refer to diagram 10, page 35.)

The input of this stage is the same array of emergent structure elements (*ESE*) that was used in step 11 (diagram 8).

- 10.1 Check the analysis definition for the user's *Connection Options* setting. If it is "replace," continue. If not, exit and return to the main procedure.
- 10.2 *Loop 1*: Perform the following steps for every element in the *ESE* array.
- 10.3 Get the next element in the array.
- 10.4 *Loop 2*: Perform steps 10.5-10.8 for every issue-impact value from 0 to 5. Deletions will be made from the bottom of the tree up.
- 10.5 Create a new array of *CNC* objects. Fill it with all existing *CNC* objects whose child records are not issues. In other words, create an array of all existing person-issue connections, as created in diagram 1, step 12.
- 10.6-10.8 *Loop 3*: Delete all the existing person-issue connections from the database.
- 10.9 Repeat for every issue-impact value from 0 to 5.
- 10.10 Get the next emergent structure element in the *ESE* array. When finished, return to the main procedure (diagram 1, step 14).

## 2.5.11 Diagram 11: Inserting new connections

The final step in the emergent structure analysis is to insert the issue-issue and person-issue connections into the database (see diagram 11, page 36).



- 11.1 *Loop:* Perform the following steps for every element in the array of issue-issue and person-issue connections created in diagram 1, step 12.
- 11.2 Get the next element in the array as an object of type *CNC*.
- 11.3 Insert the *CNC* connection in the database.
- 11.4 Get the next array element.

When all objects have been processed, the database contains a record for each issue-issue and person-issue connection in the current analysis. The display engine reads these records and adds them to the screen display (see *3.0 Display options*, page 53).

## 3.0 DISPLAY OPTIONS

The emergent structure analysis produces data consisting of person-issue connections and issue-issue connections, and stores it in a SQL database. Options for retrieving this information and displaying it fall into two main categories:

1. using a prepackaged report generator
2. creating a custom display application

### 3.1 REPORT GENERATORS

Numerous report-writing packages exist for the purpose of processing database queries and retrieving specified data. This approach requires minimal programming. Its disadvantage is that the capabilities of such packages are fixed and limited: for example, they typically produce only tabular reports and cannot display data in a graphic, hierarchical format.

Representative applications in this category are report writers such as Crystal Reports (Crystal Decisions company), those included in standard database packages from vendors such as Oracle, Sybase, IBM, and Microsoft, and numerous independent and shareware ODBC-compliant database viewers.

### 3.2 CUSTOM DISPLAY APPLICATIONS

The more useful way of processing the output data is to display them in graphical form, superimposing the connections against a hierarchical tree structure of persons or issues. This approach requires the creation of an application that can read from the database and generate instructions for creating a graphical tree structure. These instructions are then passed to a display-engine application programming interface (API). The API may be capable of rendering hierarchical output in either a two-dimensional or a three-dimensional display. The EnCompass system itself uses an API created for the purpose by the Parasol Development company (see *3.2.3 The Parasol application and display engine*, page 55).

### 3.2.1 Hierarchical data display libraries

The basic way of implementing the hierarchical display is to use a standard API or class library providing two-dimensional tree-drawing capabilities. Examples of such libraries are the following (names of developers or vendors are in parentheses):

WXP3D (The Whole Experience)

Orion3D (Gabriel Peyré)

Parasol Meta Classes (Parasol Development)

Legus3D (Branimir Karadzic)

Plush (Nullsoft, Inc.)

blaxxunContact (blaxxun interactive)

Architect III (Geometric Computing)

E3d Engine (Act 3d Interactive)

ATV tree library (Christian Zmasek)

Visual C++ (Microsoft)

Swing Jtree (Java class)

tree.hh tree class for Borland C++ (Kasper Peeters)

GTK+ (GNU PL)

PopChart (Corda)

### 3.2.2 3D display engines

A fully enhanced custom display application provides the ability to represent the relationships in a three-dimensional display. For this there are 3D analogues to the APIs already mentioned, including the following:

**Microsoft Windows:**

OpenGL (multiple vendors)

Visual C++ (Microsoft)

Direct3D (Microsoft)

Parasol Meta Classes (Parasol Development)

**Apple Macintosh:**

QuickDraw3D (Apple)

**UNIX/Linux:**

Ploticus (Seawall Group Productions)

**Multiple platforms:**

Quesa (Open Source)

### 3.2.3 The Parasol application and display engine

The custom application that provides data display and other support functions to the EnCompass system is provided by Parasol Development. The Parasol application consists of two main components (see figure 17, page 56).

The *class library* provides low- and mid-level functions for handling data structures, window objects and behaviors, application interface, and basic database access. This component is documented in the volumes *C+Objects: Volume One: Foundation Data Structures* and *C+O Class Library Foundation Data Structures: User's Guide and Reference Manual*.

At a higher abstraction level, the Parasol *meta-application* includes advanced database access, Parasol-specific user interface functions, and floating-point functions which provide the 3D calculation and transformation routines making up the display engine. Documentation for this component appears in the *Parasol Developer User Guide*.

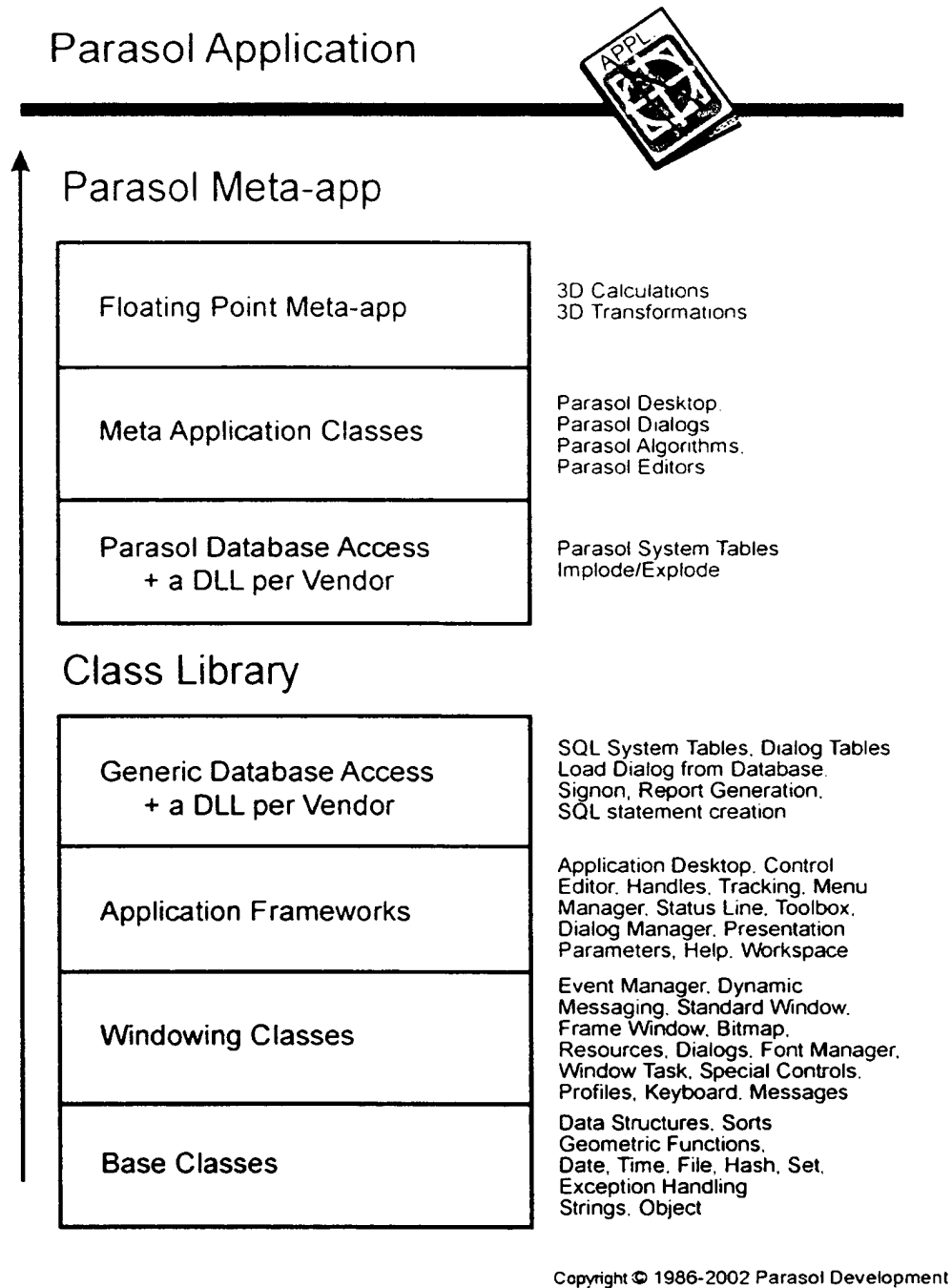



Figure 17: Parasol application structure

## 4.0 EMERGENT VIEW EXAMPLE

As an aid to understanding the concepts employed above, the following is a step-by-step example of the procedure for displaying an emergent-structure view in the EnCompass application. To follow the demonstration, install and run the EnCompass application as directed in the file README.TXT, on the installation CD/ROM.

**To display the issue view:**

1.  On the vertical tool bar at the left side of the application main window, click the "Issue" icon (or choose *Data Setup : Issue* from the *File* menu).
2. In the *Search Issue* dialog, select "Sample DB" from the drop-down list in the *Project* field, and select "Master" in the *Subtype* field.
3. Click **Search**. The program displays a list of all the master issues in the "Sample DB" project.
4. In the *Issue List*, select item IS00000004 (master issue 04, "Order Fulfillment"). From the *Search* menu, choose *Explode Down All*. EnCompass displays the *Issue View* window, containing a tree with the master issue at the top and a hierarchy of slave issues at levels 5 through 0, with the people who have been nominated to each level (see **Nomination**, page 14) attached at their respective levels (figure 18, page 58). (Note: The number of levels in the display should be set to at least 6, and the *All* box should be checked. To show the connections among people at each level, choose *Hierarchy* from the *Show* menu. Press F1 for help on display functions and other program features.)

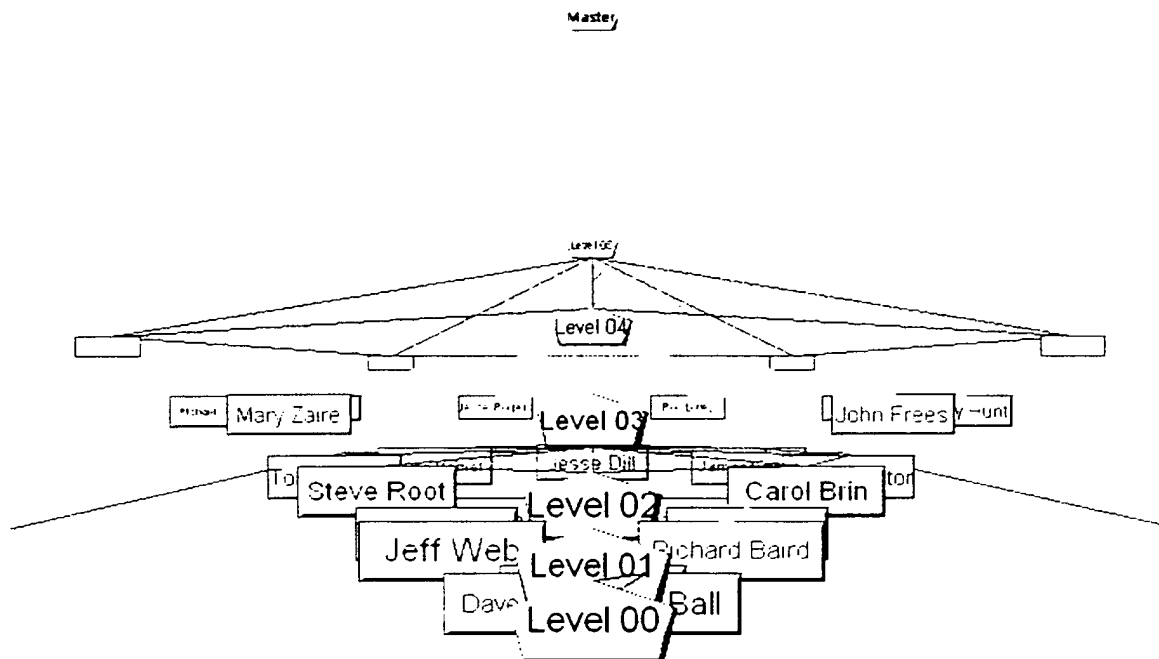


Figure 18: Emergent structure display, 3D view

For an alternative, two-dimensional view of the same tree, choose *Outline* from the *View* menu (figure 19, page 59). In either the 3D or outline view, observe that 4 names are attached at level 5, 8 at level 4, 7 at level 3, and so on.

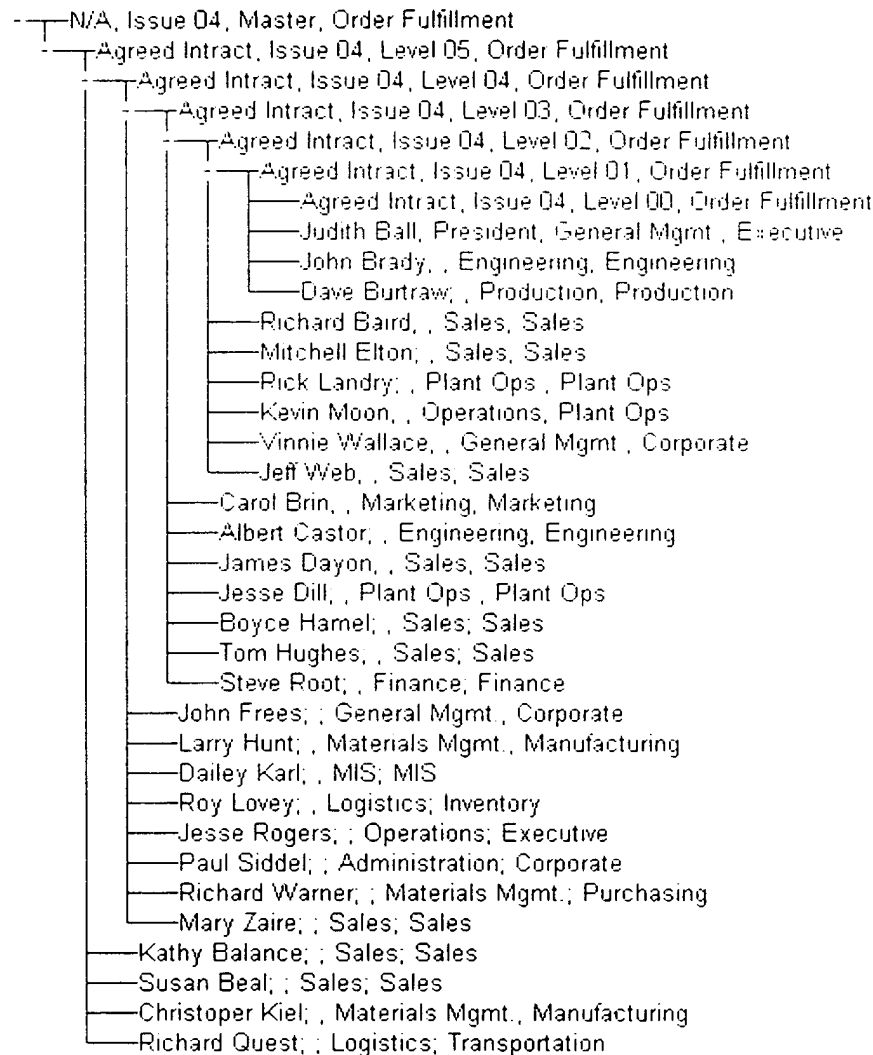



Figure 19: Emergent structure display, outline view

### ***Displaying the organization structure***

It should be remembered that the positions of these people in the issue hierarchy depend on their impact on the issue of "order fulfillment," as evaluated by the colleagues with whom they interact on a regular basis. These positions may have little to do with the individuals' status within the official organization hierarchy. To view that structure, display an *Organization View* as follows:



1.  On the vertical tool bar at the left side of the application main window, click the "Organization" icon (or choose *General Data : Organization* from the *File* menu).
2. In the *Search Organization* dialog, click **Search**. The program displays a list of all the organizations in the database.
3. In the *Organization List*, select item OR00000005, "Acme Products Inc."
4. On the *Search* menu, click *Explode Down All*. EnCompass displays the organization structure in the *Organization View* window.